National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

A RELATION MODEL FOR ANIMATING ADAPTIVE BEHAVIOR IN DYNAMIC ENVIRONMENTS

BY

Hanqiu Sun

©

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL 1992

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-77401-0

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **Hanqiu Sun**

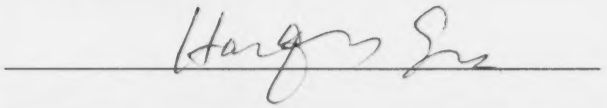
TITLE OF THESIS: **A Relation Model for Animating Adaptive Behavior in Dynamic Environments**

DEGREE: **Doctor of Philosophy**

YEAR THIS DEGREE GRANTED: **1992**

PERMISSION IS HEREBY GRANTED TO THE UNIVERSITY OF ALBERTA LIBRARY TO REPRODUCE SINGLE COPIES OF THIS THESIS AND TO LEND OR SELL SUCH COPIES FOR PRIVATE, SCHOLARLY OR SCIENTIFIC RESEARCH PURPOSES ONLY.

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION.



PERMANENT ADDRESS: Cha Gang Xin Cun #31
Shui Guo Lake
Wuhan, Hubei, China
430071

DATE: **September 4, 1992**

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

THE UNDERSIGNED CERTIFY THAT THEY HAVE READ, AND RECOMMEND TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH FOR ACCEPTANCE, A THESIS ENTITLED **A Relation Model for Animating Adaptive Behavior in Dynamic Environments** SUBMITTED BY **Hanqiu Sun** IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF **Doctor of Philosophy.**

Barry Joe
Professor Barry Joe (Supervisor)

Wm. Armstrong
Professor Bill W. Armstrong

Hong Zhang
Professor Hong Zhang

Allen Carlson
Professor Allen Carlson

Brian Wyvill
Professor Brian Wyvill, University of Calgary

DATE: Sept. 4, 92

To My Parents

Abstract

For nearly 20 years, techniques for animating the motion of a single object have been studied. When this problem domain is extended to dynamic environments, which involve multiple moving objects and unpredictable scene events, the animation problem becomes more complicated. There are three reasons for this increase in complexity. First, interactions between objects and between objects and the environment increase the number of degrees of control in a motion. Second, unpredictable environmental influences on the motion make it impossible to pre-plan the details of the motion, and the object must be able to react to these events when they occur. Third, detailed control or pre-planning of a behavior makes it difficult to explore similar behaviors or environments and makes the editing of the motion difficult.

Current animation research attempts to deal with the scene motion problem in the same way that a single object's motion is produced. This approach codes the motion in one predefined sequence, which is hard to reuse for slightly different environments. Other approaches proposed for this problem are the sensor-effector approach, rule-based approach, and predefined environment approach. These approaches, however, are either incomplete or limited to certain behaviors or simple environments.

This thesis proposes a relation control model for the problems of animating motion in dynamic environments. Using this approach, a scene is first decomposed into a set of relations, each describing one environmental influence on the motion. These relations are modeled on a one-source-one-responder basis. The relations are used as control primitives to build up a control hierarchy that produces the desired behavior. Each level of the hierarchy represents one aspect of behavior, starting from the selection of relations, and working up to relation interactions, elementary behavior patterns, and behavior sequences.

This thesis discusses relation definitions, analysis of relation classes, objects overlapping

among relations, and estimates of the minimum and maximum number of relations required for certain motions. The general modeling frame for relations and a four-level hierarchy for behaviors is outlined in detail. Examples using both the frame modeling and the hierarchical level structuring are presented. Examples of using the system's facilities have shown the potential of the relation model for allowing the free expression of behaviors in an environment.

The relation model is an alternative and better approach for dealing with the special problems introduced by dynamic environments. It reduces the complexity of motion control by decomposing it into atomic units (relations) and providing a hierarchical mechanism for combining these units. It directly links the environmental influences to the relations, each of which is only active when a particular condition is sensed in the environment. The behavior hierarchy allows the simple editing of each level of the behavior, while exploring other similar behaviors and environments. As one of the first solutions to this problem, the relation model opens a new dimension for the study of environmental issues, extending from the boundary of a single object's motion to dynamic environments.

Acknowledgements

This thesis would not be possible without the critical review and helpful comments from my supervisor, Dr. Barry Joe. I am also indebted to Barry for his kindness, responsibility, and effort for guiding the final approval stage of this thesis work.

Special thanks go to my examination committee members, Dr. W. Armstrong, Dr. H. Zhang, Dr. A. Carlson, and Dr. B. Wyvill for their valuable comments on the quality of the work, especially the detailed comments and suggestions from Dr. B. Wyvill, my external examiner.

Special thanks also go to Dr. M. Green, my former supervisor, for his initial interest in the work, constant encouragement during the work, many helpful discussions, and patient reading of the early drafts of this thesis.

Many thanks are due to the people working in our graphics lab, C. Shaw, H. Wang, Q. Tang, Y. Sun, and J. Leung, for the good working environment they create in our lab and their useful discussion, encouragement, and support during this work. Thanks also go to our weekly graphics meeting for exchanging the ideas and sharing the working experience from each of us.

Thanks also go to the support group in our department for maintaining such good running facilities for us to use and many great help for solving every occurrence of hard crisis. Special credit goes to C. Smith and R. Lake for their patient help for the preparation of this document.

Finally, I would like to express my sincere thanks for my parents for their belief in me and their endless moral support of the work. Thanks also go to many of my friends, whose names are not mentioned here, for their care, friendship, and encouragement that have definitely contributed to the work.

Table of Contents

Chapter	Page
Chapter 1: Introduction	1
1.1. What Is Computer Animation?	1
1.2. Principles and Techniques Used in Traditional Animation	3
1.3. The Roles of Modeling and Rendering in Animation	7
1.3.1. Modeling and Motion Control	8
1.3.2. Rendering and Motion Control	11
1.4. The Problems and Research Goals	12
1.5. Outline of the Thesis Chapters	16
Chapter 2: Research Trends in Computer Animation	18
2.1. Groups of Animation Research	18
2.2. Kinematics vs. Physically Derived Motion	20
2.2.1. Kinematics	20
2.2.2. Physically Derived Motion	23
2.3. Interactive Tools vs. Programming Language	28
2.3.1. Interactive Tools	28
2.3.2. Programming Language	32
2.4. High-level Abstraction vs. Multi-level Interfaces	36
2.4.1. High-level Abstraction	36
2.4.2. Multi-level Interfaces	38
2.5. Approaches Using the Animation Techniques	39
Chapter 3: Problems Review and Research Outline	41
3.1. Review of Problems	41
3.1.1. Problem Size -- Number of Degrees of Freedom	41
3.1.2. Implicit Behavior Modeling Structure	45
3.1.3. Indirect Environment Control Effect	46
3.2. Research Motivations	48
3.3. Previous Approaches	51
3.3.1. Sensor-effector Approach	51
3.3.2. Rule-based Approach	53
3.3.3. Predefined Environment Approach	56
3.4. Proposed Relation Approach	58
3.5. An Example Using the Relation Approach	60
Chapter 4: Theoretic Foundation of the Relation Model	64
4.1. Definitions of the Relation Model	64
4.2. Special Control Properties of Relations	72
4.3. Related Objects Mapping	79
4.4. Class Types of Relations	84
4.4.1. Static Relation Sources	86
4.4.2. Dynamic Relation Sources	86
4.4.2.1. Moving Object Sources	87
4.4.2.2. Scene Event Sources	88
4.5. Minimum and Maximum Number of Relations	90
Chapter 5: Outline of the Relation Model	94

5.1. Frame Modeling for Relations 94
5.2. Hierarchical Structuring Mechanisms 103
5.2.1. Previous Research on Control Hierarchies 105
5.2.2. Relation Composing Hierarchy 107
5.2.2.1. Selective Control 109
5.2.2.2. State Control 111
5.2.2.3. Pattern Control 113
5.2.2.4. Sequential Control 115
5.2.3. Other Suggestions 116
5.3. Relation Processing Algorithm and Its Complexity 117

Chapter 6: Prototype Implementation 122
6.1. System Architecture 122
6.2. Objects and Relations Scripting Language 127
6.3. Interactive Control System 132
6.3.1. Scene Composing Component 135
6.3.2. Pattern Control Component 136
6.3.3. Sequential Control Component 140
6.3.4. Scene Rendering Component 141

Chapter 7: Examples of Dance Motion in Various Environments 145
7.1. Dancer and Dance Notations 145
7.2. Dance Motion in Static Environments 150
7.2.1. Static Environment with a Group of Obstacles 150
7.2.2. Static Environment with Other Individuals 154
7.3. Dance Motion in Dynamic Environments 157
7.3.1. Dynamic Environment with Other Moving Objects 158
7.3.2. Dynamic Environment with Additional Dancers 161
7.3.3. Dynamic Environment with Scene Events 164

Chapter 8: Conclusions 167
8.1. Research Summary 167
8.2. Major Contributions of the Research 171
8.3. Future Work 173

References 175

Appendix A1: Interactions in the System's Components 180

Appendix A2: Modeling of Relations in Various Environments 189

List of Figures

Figure	Page
1.1 Squash and Stretch in Bouncing Ball	3
1.2 Shape Deformations	4
1.3 Multi-plane Animation	6
2.1 In-betweening Using Moving Point Constraints	21
2.2 Forces and Torques Acting on the Body	24
2.3 A 14 Legged Insect in Its Walking Cycle	31
2.4 Interactive Setting of the Goals of the Lower Torso and the Hand	31
2.5 The Class Hierarchy in the Modeling Portion	35
3.1 Sensor-effector Control Model	52
3.2 Rule-based Control Model	54
3.3 Predefined Environment Control Model	56
3.4 Relation Control Model	58
3.5 Relations Interrupting Scheme While Walking in a Room	63
4.1 A Response Between the Two Related Objects	68
4.2 Four Mapping Types of a Relation	80
4.3 Three Primitive Patterns Among Overlapped Relations	83
4.4 Class Division of Relation Types	85
4.5 Reflective Responses Between Two Object Groups	87
4.6 Example of Relation Mapping Graph in an Environment	90
5.1 Three Control Sections of Relation Frame	94
5.2 Simplified Version of Relation Control Frame	102
5.3 Levels of Control Hierarchy in the Skeleton Animation System	105
5.4 Levels of the Barr's Hierarchy	106
5.5 Levels of the Relation Composing Hierarchy	109
5.6 Three Structures for a Possible State Transition	113
5.7 A Possible Ordering of Three Behavior Patterns	115
5.8 Grneral Relation Processing Algorithm	118
5.9 Block Diagram of the Relation Control Process	118
6.1 Input and Output Process Interface for RCS	123
6.2 Structure of the RCS System	126
6.3 Syntax for Standard Object Types	129
6.4 Syntax for the Procedural Object Type	129
6.5 Menus Hierarchy for the Four Interactive Components	132
6.6 Screen Division of the Four Interaction Areas	133

6.7 Internal Data Structure of the Relation Table 138
6.8 Internal Data Structure of Patterns 140
7.1 Tree-like Structure of Human Figure's Model 145
7.2 Interactive State Structures of Pattern P1 and P2 153
7.3 Revised State Structures by an Additional Individual 156
7.4 Interactive State Control of the Jumping Motion 159
7.5 Revised State Controls with Other Moving Object 160
7.6 Revised State Controls by Two More Dancers 162
7.7 Revised State Controls with the Event Occurring 165
8.1 System Boundary Between Specification Time and Run Time 172

Chapter 1

Introduction

Computer animation is the study of natural computer models for representing the real and imaginary dynamic world. Whether current computer technology can reproduce natural objects and their motions has been an open research problem and a great motivation for animation research. For a better view of animation research, the concept of computer animation and its potential applications in other closely related fields, modeling and rendering, are first introduced. A brief discussion of the general problems in animation and the research goals of this thesis are then presented. Following that, the chapters of the thesis are outlined.

1.1. What Is Computer Animation?

The initial fundamental theory of computer animation comes from a well known characteristic of human vision called the persistence of vision. This refers to the retina's ability to retain the image of an object for a brief instant after the object has been moved. Thus, a series of still images presented in rapid succession produces an illusion of motion. If the still images, also called frames, depict the progressive phases of a single movement, the eye will perceive them as one continuous flow of motion. Typically 24 frames/second are used in film, and 30 frames/second are used in video.

Computer animation uses computer based techniques to generate and manipulate sequences of moving images. A number of properties of a modeled object can be dynamically manipulated over a period of time, and these variations are not merely restricted to physical movement. It could be a change of color, light intensity, camera location, character emotion, or shape. For example, camera motion is manipulated for flight simulation using a parameterized camera model. Moving scenery along a flight course is produced by changing the position and orientation of a viewer's eye. Another example involves the mouth movement resulting from the process of shape deformation, based on the word being said and the continuous transition between words. This deforming process requires more complicated control than scaling or

rotating the geometrical primitives of the mouth.

In practice, computer animation involves geometry, graphics rendering, motion control, video synthesizers, image processing, computer assisted cartoon animation, special effects and three dimensional image synthesis. Each part plays a role to complete or improve the quality of a motion sequence. The range of operations can include camera projection, lighting effects, editing the pixels in an image, or geometric subdivision of an object model. Thus, computer animation is the process of controlling a set of variable properties over time and the recording of the results of these changes as a sequence of visual images. The time reference of animation, as the fourth control dimension, provides an essential clue to the understanding of these changes.

Motivated by nature, some research in computer animation has focused on reproducing the motions that occur in the real world. These include the swaying motion of a tree or trees in the wind, and the impact effect of two cars colliding with each other. The motion can be naturally generated if it is fully understood. In practice, the animation of natural motions has potential applications in the fields of education, scientific visualization, spacecraft simulation, medical surgery, and others, where the realistic effects of motion is essential.

In addition to the applications of realistic animation, computer animation has been used in areas where a sequence of moving images is required to illustrate the actions of a dynamic process. Examples of such applications are algorithm animation and command animation in a graphical user interface. It can be more intuitive to view the dynamic process of sorting an array or updating a tree structure by a sequence of animated images. The semantics of a copy command can be identified from the icon of the copied material moving towards its goal location, as animated in a graphical user interface. These examples show another purpose of computer animation -- the illustration of dynamic control processes over time.

1.2. Principles and Techniques Used in Traditional Animation

From over 50 years of experience, the production of interesting motions has been reduced to a few fundamental principles of animation. These include squash and stretch, overlapping action, ease in and ease out, exaggeration, anticipation, and secondary action. These fundamental principles have guided animators to produce better motion over the past decades. Although more advanced animation techniques have since been developed, these traditional principles are still suitable for producing better animation. One example of using the basic principles in today's computer animation is the Luxo Jr. animation [Lasseter87], produced by the animation group of Pixar in 1987.

The object (or person) being animated will undergo certain changes within its overall shape. For example, squash and stretch techniques [McQueen87], which represents the elasticity of an object, implies a sense of weight, mass or other physical quantities. This motion effect can be easily supported by global scales which keep the volume of the object constant throughout the scene. For example, if a sphere is scaled by 0.5 unit along its height to imply a squashed ball smacking against the ground, an appropriate scale, say 1.5, units along its width would be necessary to keep the implied volume constant. An example of squash and stretch is the motion sequence of a bouncing ball shown in Figure 1.1.

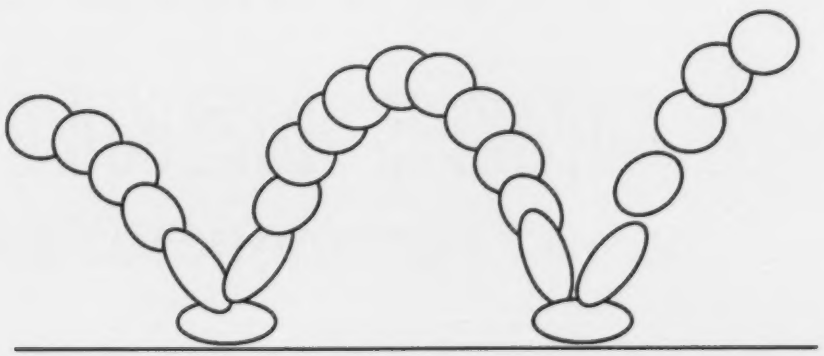


Figure 1.1 Squash and Stretch in Bouncing Ball

Overlapping action [McQueen87] is another traditional method to minimize the abrupt changes during transitions between actions. A new action could start before the previous action is brought to a complete stop. In effect, a continual flow and continuity between whole phases of actions are maintained by overlapping. Consider a person falling down to the ground; it is natural to keep the person's arms and hair up for a few frames while the falling action overlaps with the landing action.

Ease in and ease out [McQueen87] are usually used at the beginning or end of an action to 'soften' the transition between active and static states. A fluid action is greatly enhanced with a gradual speeding up or slowing down of a motion. Also, ease in and ease out can visually simulate the internal physical state of objects and external strength of a motion. A heavy object has a slower ease-in than a lighter one, and a ball rolls much faster if it is pushed hard.

In addition, exaggeration, anticipation and secondary action [Lasseter87] refer to subtle and natural visual effects which help support the main action in a scene. For example, when the lamp, Luxo Jr., jumps up for a hop, his whole body movement is exaggerated to give the feeling of realistic weight to his base. When the lamp lands after a hop, the impact is shown in the exaggeration of his body movements. A zooming off action of a character is anticipated by its drawing back like a spring in the opposite direction of motion. Someone sitting might have secondary actions such as tapping their toes or drumming their fingers against the arm of a chair. These classical principles used in 2D hand drawn animation are still used in creating comprehensive computer generated animation in today's systems.

In traditional animation, an animation sequence consists of a number of drawings, each of which is produced by hand and recorded on film in the appropriate time order. The main parts of the motion are prespecified in keyframes, and the rest of the frames are produced by linear or spline interpolation between the keyframes. The key drawings do not make up an entire action, but they are sufficient to guide the inbetweener from one position of the character or object being animated to another. One example of shape transition between two key positions is shown in Figure 1.2. Here a key-hole shape changes to a circle.

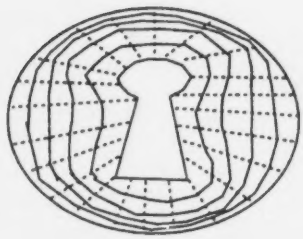


Figure 1.2 Shape Deformations

A color table has been used to produce limited animation through clever manipulation of colors on a static display. Without actually moving any shapes (the shapes already exist in the memory of the display device), different versions of the object appear on each frame by changing the colors. Practically speaking, color table animation is a low cost technique, but often completely satisfactory for many applications as only the color content is changed over time. One example of color table animation is producing a sequence of balls from the smallest one to the largest one, where the action is created by sequentially illuminating shapes that are already defined.

Another technique used in traditional animation involves organizing an object's parts into layers according to the dynamic properties of the parts. For instance, a human figure can be divided into three layers of head, body and arms, and background as shown in Figure 1.3. The division into layers can also be applied between objects, for example using two layers, one for foreground and one for background. The foreground separation from the complex and less movable background simplifies the description of the motion. Two-dimensional animation with a three-dimensional background is a traditional approach that has been used in producing Disney style cartoon animation. Decisions on inbetween keyframe interpolation at different layers, such as using cubic spline interpolation for foreground and linear interpolation for background, increase production flexibility. On the other hand, layer animation restricts the motion dependency between layers. For the previous example, a strong push on the shoulder may not only move the figure's

body and arms, but also pass the movement up to the head, or down to the legs, since the parts are connected. If the head and body are on different layers as shown in Figure 1.3, this movement cannot be easily propagated in a realistic way.

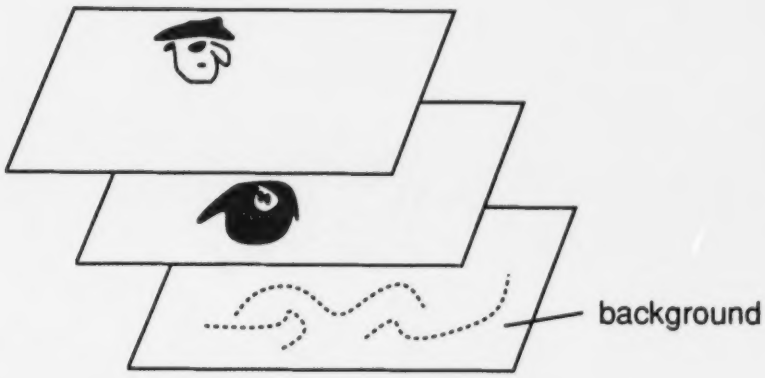


Figure 1.3 Multi-plane Animation

As the motion is mostly animated by hand, the use of conventional techniques cannot avoid several drawbacks. First, the control of motion is limited to a very low level by hand drawing. Whether a realistic motion can result or not depends on the animator's drawing skills. In other words, the motion is only controlled by detailed drawing of every frame. Second, motion is passively derived from an initial position to a final goal. If a final condition is not met, the sequence of frames must be redrawn over and over again. It is painful to correct a final pose once the sequence has been completely drawn. Third, the range of motion is bound by two dimensional hand drawing. This is the reason why early animation is dominated by 2D cartoon motions using kinematic transformations.

The challenge to provide the animator with better tools in a flexible system environment has motivated animation research from its beginning. Many advanced control techniques have been proposed and developed in various systems. Among them, the detailed controls of motion are separated to multiple levels of a control hierarchy. At the highest level, a motion can be issued by a command such as "walk to

the door" and the system fills in the control details required for such a motion. In contrast to forward driving, a motion can be inversely derived from a goal description to the initial control conditions. Not restricted to 2D cartoon images as in traditional animation, realistic three dimensional images and motions are modeled using the techniques of dynamics, motion scripts, procedural description, and natural language. A detailed discussion of the advanced research in this area is presented in Chapter 2, and some of the remaining problems and research goals for this thesis are outlined in section 1.4.

1.3. The Roles of Modeling and Rendering in Animation

Computer generated motion sequences are basically processed by three consecutive steps: modeling, motion control, and rendering. The modeling step builds the geometrical form of objects either by programming or interactive modeling based on the primitive parts provided by the modeling system. Once modeled, these objects are then manipulated to form a series of frames, one at each time step. Finally, the sequential frames are rendered using light models and other visual effects and projected from the three dimensional space to a two dimensional view plane.

The process of motion control relies on the other two parts: modeling and rendering. Motion control manipulates the geometric form of objects defined by modeling, and the manipulated objects in each frame are illuminated by rendering. However, modeling and rendering are not completely separate processes from the motion control part; in many aspects, the concepts and techniques developed to control a motion can be utilized in modeling and rendering. Examples of combined processes are the modeling of a fractal mountain and the rendering of a camera model or color table. Since there is an overlap between these three processes, an introduction to the basic models and well-known techniques used in modeling and rendering, as well as their interaction with motion control, are briefly reviewed in the following subsections.

1.3.1. Modeling and Motion Control

Modeling techniques can be divided into a few classes: polygons, curves and surfaces, fractals, and particles. These classes are identified by a set of common control properties, that determine the possible dynamic behavior produced by objects in the class. The polygon class is suitable for regularly shaped objects such as bridges and buildings. Surface oriented objects like tori and teapots are best modeled by the curve and surface class. Other natural phenomena like clouds, terrain, and fireworks are modeled by fractals or particles, using stochastic processes.

Manipulations for polygon objects are restricted by the fact that the polygons must be flat. Usually the motion of polygon objects is generated using a composition of scaling, rotating, translating, and shearing transformation matrices. Two examples of polygon animation are Burtnyk's stickman figure and Parke's facial model. The stickman figure [Burtnyk76] is modeled by polygon segments which form parts of the body, moving relative to each other at joints through a transformation hierarchy. The facial animation by F. I. Parke, at the New York Institute of Technology [Lewis87, Parke75, Parke82], is based on a hierarchical polygon network. Complex facial expressions such as smiling and frowning are controlled by grouped parameters in the underlying polygon structures. However, generally speaking, the possible manipulations of polygon objects are very simple, predictable, and thus unsatisfactory in most realistic animations because of missing details.

A surface patch or a curve segment is essentially defined by a set of control points. When combining several surfaces and curves, the first derivative continuity (and sometimes the second derivative), can be adjusted at the common boundary in terms of a few related control points. The objects modeled in the curve and surface class can be selectively modified either in a local area or at a global level. This property has motivated research into providing flexible modeling control of curved objects using interactive modeling systems. Among the various types, Bezier, Coons, and B-splines are the most commonly used curve and surface types in practical models.

The particle and fractal modeling classes use stochastic processes to model random behavior. The techniques developed in these two modeling classes can cover a broad range of natural phenomena, such as clouds, terrain, trees, and fire. Even though the two classes have very similar motivation for modeling natural objects, the classes have different approaches to applying the stochastic process.

Particle systems, introduced by W. Reeves [Reeves83], is a method for modeling a class of fuzzy objects such as fire, clouds, and water. These systems generate a cloud of primitive particles over a period of time. These particles move and change form within the system, and eventually die from the system. All changeable attributes of a particle, such as position, velocity, size, color, transparency, shape and lifetime, are stochastically determined by mean values and maximum variations. One example of computing a particle's speed is

$$Speed = MeanSpeed + Rand() * VarSpeed$$

Particle systems build complex pictures from sets of simple, volume-filling primitives. For large numbers of three-dimensional particles, a new rendering technique [Reeves85] based on approximate and probabilistic algorithms has been proposed to reduce their rendering cost.

Fractal systems [Kajiya82, Kajiya83, Norton82] model natural objects by successively adding details to a simple primitive, such as triangles or quadrilaterals commonly used in two or three-dimensional fractals. An overview of the combined deterministic and stochastic controls of some primitives and their motions is provided by Fournier et al. [Fournier82], based on the theoretical work of Mandelbrot [Mandelbrot75-Mandelbrot82]. Basically, the fractal approach recursively subdivides a primitive to produce more details by connecting the subpolygons from the midpoints of the sides, where the midpoints are randomly varied within some limits. This subdivision idea has been applied to one-dimensional fractal lines, two-dimensional fractal planes, three-dimensional fractal surfaces, and four-dimensional fractal motions, such as a leaf in the wind. More complex models like a mountain or terrain can be similarly subdivided from a coarse mesh of triangles or quadrilaterals. One advantage of using fractal modeling is that possibly unlimited amounts of object detail can be automatically generated based on few control parameters.

However, fractal systems require computer time to recursively generate details.

Models that combine several different techniques have attracted research attention. Smith [Smith87] discusses the common control properties of different techniques used for producing natural phenomena, in particular tree-generating systems. The systems implemented by fractals, graftals, and particles are compared with examples of tree generation. Unlike the geometric grammar used in a fractal system, a graftal system uses a topological grammar to describe the growth pattern of trees, which in turn is interpreted geometrically at drawing time. Similarly, Prusinkiewicz et al [Prusinkiewicz88] use L-systems for modeling plants.

Another example is the modeling system proposed by Barzel and Barr [Barzel88], where constraints are applied to the modeling of objects. The system specifies the geometric constraints of "Point-to-Nail", "Point-to-Point", "Point-to-Path", and "Orientation" as relations between primitive elements. To satisfy these constraints, a set of dynamic forces are found and used to guide the motion of the elements. Moreover, the elements not only move towards their destinations defined by geometric constraints, but also maintain positions and orientations which are constrained in a dynamic control environment. This enables the modeling task to assemble the elements of a rigid object by controlling the temporal behavior of the elements.

The use of mathematical constraints for the dynamic modeling of physically-based flexible models [Platt88] is another combining technique. The properties of reaction constraints and augmented Lagrangian constraints are used in the flexible models to simulate the effects of moldability and incompressibility. These properties can characterize the dynamic behavior of flexible models when they interact with other objects in the environment. These interactions can be very difficult to produce using traditional modeling techniques. Platt's approach allows the control of reaction constraints to guide flexible models along a path using a reduced amount of computing time to detect object collisions, while augmented Lagrangian constraints can maintain the models with volume-preserving squashing and molding of taffy-like substances. By using these techniques, interesting scenes such as a sphere squashing a seat cushion and a compressible

gelatinous cube hitting a table have been modeled in a physically realistic way.

1.3.2. Rendering and Motion Control

Temporal anti-aliasing, also called motion blur, is a research topic involving rendering and motion control. Motion blur is a visual effect caused by the object's movement during the exposure time of a camera and is often used to give the viewer the illusion of the motion when the object or camera moves rapidly. A technique for modeling the effects of a lens and aperture in a virtual camera is presented by Potmesil and Chakravarty [Potmesil82]. In this technique, motion blur is generated by convolving moving points with the optical system transfer functions in terms of the path, object's velocity, and exposure time of the camera. Alternatively, the supersampling approach proposed by Korein and Badler [Korein83] uses multiple intensity buffers, each one corresponding to a different point in time.

The animation of cameras [Karp90] introduces rendering issues to research in motion control; these include camera motion planning, material and viewing style selection, visual continuity control, and multiple viewport settings. The setting of an additional viewport can depict the details of local interest from the main viewport; the same event or simultaneous related events can be presented in two or more perspective views. With a better organization of cameras, a motion can be assisted using some of the filmmaking techniques and knowledge from the animation domain.

For each frame of a motion sequence, the rendering process usually requires considerable computing time for realistic effects. The cost of realistic rendering can be traded off for various rendering details. If using an interactive animation system, the system can display objects as wireframes for interactive playback to minimize the high rendering cost, or all frames in an animation sequence can be rendered off-line by another process.

1.4. The Problems and Research Goals

The essential research issue in computer animation is not how to generate a motion, but how the motion can be generated more naturally and easily. In traditional animation, a motion is generated by hand drawing from one frame to the next and edited in a "draw-view-redraw" cycle. It is possible to produce an animation frame by frame even using this primitive drawing method and its redrawing cycle. However, drawing by hand, animators work in a very low level control environment in which they have to specify all the control details in each frame of a motion. If a small positioning error is detected or a slight change is made when reviewing a generated sequence, the sequence has to be redrawn.

This limited control environment is ever more difficult when animating a motion with some dangerous actions that can not be experienced, or when there is a high degree of interaction between objects that is not easy to predict. This difficulty can double or triple the number of the redrawing cycles. The addition of these two -- drawing and redrawing, makes an animation task extremely expensive and time consuming.

Even though the drawing technique used in the traditional animation has been replaced by computer techniques, limited control of animation to some degree still remains. To discuss the issues, we first look at the main differences that have resulted from using today's computer techniques. From the three drawbacks outlined in section 1.2, the two dimensional drawing plane of traditional animation has been replaced by the three dimensional modeling space; a sequence of drawn frames is now produced by a computer-oriented control system; and the primitive drawing has been extended to various hierarchical control levels.

A motion can be specified by programming (or some other high-level control means such as using a dynamic control package or a control algorithm) and edited in a "program-execute-view-reprogram" cycle. A motion generated by a computer can be edited through an interactive interface at run time. However, this type of editing is currently limited to only a few predefined control parameters, which can not satisfy the general needs of changing an existing motion.

The motion generated by the programming and testing cycle can be expensive and time consuming. The remaining difficulty mainly comes from the testing cycle, even though the cycle has changed from hand drawing to computer programming. For instance, for a few seconds of animation, animators must program all the control details for each frame of the motion. If some error occurs when reviewing the generated motion, the motion needs to be reprogrammed, recompiled, and reexecuted. This trial and error cycle may need to be repeated many times before producing the desired motion. Especially for animating a complex motion involving a large control degrees of freedom and dynamic interactions between moving objects, it is not clear in the first place how the motion should be programmed. In this case, many tests are required to match the correct motion image that an animator has in mind to the program code.

For the remaining problem, two solutions have been currently proposed: high-level languages and interactive systems. High-level languages, including animation languages, natural languages, and constraint languages, can reduce some of the burden of debugging an animation using a simplified descriptive language tool. Without worrying about every low-level control detail, the user can better concentrate on the problem and describe the problem at a level that is closer to our natural understanding. This means that fewer errors will be made while describing the motion. On the other hand, the high-level specification ability is restricted by the assumptions made to support such a language, as well as the cost of designing the language. The same test cycle remains for using the high-level languages.

The use of an interactive animation system provides a good testing cycle for varying the control of a motion. Using an interactive system, the user can interactively edit the motion and view the change instantly on the screen. This on-line testing environment reduces the length of the testing cycle "what you see is what you get". However, the use of an interactive system restricts the motion that can be modeled with the system. The ability to vary the motion is limited to a set of parameters or the use of a few commands which automatically generate the motions previously defined by the system. In this case, the user has little to say about how a motion behaves. The instant change ability provided by an interactive system is mainly limited to a set of parameters, which varies the strength of similar behaviors. Some systems allow

the user to interactively specify a set of keyframes for a motion, which are then automatically interpolated to a smooth motion by the system. This type of system needs the support of highly interactive 3-D input and output devices, which are currently in the research stage.

This research proposes a new motion control model that uses neither a complete programming language nor an interactive system limited to a few predefined motion cases. Instead, this new model combines the two approaches to form a better system control environment. New control concepts and mechanisms are introduced in this environment, for specifying atomic controls and structuring these controls to complex motions. Using the environment, the user can interactively compose a motion by coordinating the interactions between objects, scheduling events and their responses, and defining the unique behavior styles. With the ultimate goal of reducing the limitation of the control environment, especially the testing cycle, this thesis proposes an alternative and better approach for generating dynamically changing motions, in an intuitive, creative, and structured system control environment.

The intuitive control environment refers to the new concept of control introduced by the relation model. This concept is based on the idea of decomposing the motion into primitive units. That is, instead of generating a motion step by step in a sequential order, the possible elementary controls which could be used in a motion are first described (by programming or scripting). Each of these primitive units matches our natural understanding of simple control effects. In this way, the task of describing an animation is simplified into smaller problems. These elementary controls serve as the very basis for modeling complex motion behaviors. At some higher level, possibly using an interactive environment, the user can freely build some constraint network or tree structure among these elementary controls. The interactive system for specifying the interconnection should be natural for the user.

The creative system control environment refers to a highly interactive environment. This environment should not restrict the control parameters to a predefined motion. Instead, the user's passive role in interactively repeating the motion should be changed to an active one. The user should be able to interactively define and modify a motion both locally and globally. The system should not place any constraint on

how a motion is generated, but should let the user use his or her creative ability to generate the desired motion. The motion generated by the user may not be previously known to the system. A debugging environment is supported to interactively model and edit the motions through the system's interfaces.

The structured system control environment refers to the structures which can be modeled among the elementary controls. These structures are used as the high level control mechanisms used to produce complex behaviors. Each behavior may depend on the users, dynamic situations, and experimental purposes. These different behaviors can be built from the structures modeled among the elementary controls used in the motion. Thus, a structural control editor can be used to flexibly model and revise an object's motion. The use of this structure editor allows for exploring alternative behaviors, without even touching the detailed code.

The animation model proposed here is mainly aimed at the middle levels of the motion control hierarchy. The lower levels of the hierarchy address the low level control details, such as computing inbetween frames given a set of keyframes, or computing the motion of an object given the forces and torques that are acting on it. Towards this purpose, our model decomposes the control complexity of programming a complete motion into separate modules. Our model also suggests hierarchical structuring mechanisms that can be used to produce complex motions. With these explicit structuring mechanisms, a complex motion can be modeled with great ease. Different motion behaviors can be flexibly tested by varying the structures built in the motion. Our model does not specify the low level control details of a motion, but given the low level control procedures our model generates the input to these procedures that will produce the desired motion.

1.5. Outline of the Thesis Chapters

The next chapter outlines animation techniques and systems based on them. Three research trends are reviewed. These are: control approach, control facility, and control hierarchy. Control approach motivates the research, control facility applies the ideas to technical details, and control hierarchy introduces possible hierarchical structures when applying the ideas. Within each trend, two major groups have been developed over the years. These include kinematics and physically-derived motion in the control approach, interactive tools and programming languages in the control facility, and high level abstraction and multi-level interfaces in the control hierarchy. Besides the research review, specific problems in applying these techniques are also presented.

Chapter 3 describes the problem of behavior animation, especially in the domain of dynamic environments. The unsolved problems in current animation research motivates the new control model of behavior animation presented in this thesis. In comparison with the new model, other approaches previously used for the problems are summarized. These include the predefined sequence approach, sensor-effector approach, rule-based approach, and predefined environment approach. In addition, the new relation model proposed by this thesis is introduced, along with an example using the basic ideas in the model.

Chapter 4 defines the theoretical foundation of the relation model. The concepts of environments (static and dynamic), an enabling condition, a response, a relation, directed graph of linked relations, and a sequential behavior are formally defined. The special control properties of relations used in dynamic environments are discussed. Relation mapping among the objects in a general environment, relation classes, and the minimum and maximum number of relations which are required in a scene animation application are also systematically outlined.

Chapter 5 presents the technical details of the relation model. These details are divided into two parts: relation modeling and relation structuring. In the modeling part, a general relation frame is used for modeling the relations of various types. The frame consists of three control sections: control header, action body, and finalization. The syntax rules for modeling a relation's control properties and its behavior are described

in detail. In the structuring part, a four-level relation composition hierarchy is proposed. These four levels are: selective, state, pattern, and sequential. Details of the structuring of relations at each of the levels are presented. Also, a general algorithm for processing the relations at each step of the animation is outlined, together with the time complexity for running the algorithm.

Chapter 6 describes a prototype implementation of the relation model. The system architecture for customizing the technical "skeleton" of the relation model and structuring its parts is outlined. Details of using a textual, frame-like language for specifying the relation frame are presented. An interactive behavior editor is introduced. Four components of the editor, for scene composition, pattern structuring, sequence structuring, and scene rendering, are described in terms of external interfaces and internal data structures of the system.

Chapter 7 presents a set of dance examples in various room environments. These examples are incrementally modeled from an environment consisting of only a few static objects to environments with more static and dynamic objects. Each of the examples is modeled from composing the environment to structuring the relations which are necessary for modeling a desired behavior in the environment. The modeling of the relations and the steps in structuring the relations for each of the examples are described in detail.

Chapter 8 reviews the important ideas proposed in the relation control model, summarizes the major contributions of the research, and comments on possible future work.

Chapter 2

Research Trends in Computer Animation

The diversity of animation research comes from the many aspects of this field, such as the object types, movements, tasks, and behaviors. Take object types as one example. If an object is made of rigid materials, the object's motion can be produced using a hierarchical model with transformation matrices. If an object is made of curved soft materials, the object's motion can be generated by changing the control vertices or skeletons of the surfaces used to model the object. This chapter discusses the control techniques and systems developed in animation over the past decades in three groups. The basic ideas, research progress, major contributions, and typical techniques and systems in each of the groups are outlined. And the important applications using these techniques are briefly reviewed.

2.1. Groups of Animation Research

To provide a clear view of animation research, the diversity of the research is divided into a few problem domains, each of which covers a set of similar problems. For instance, physically derived control is one of the domains which can be identified. In this domain, motion is controlled by using forces, torques, mass, inertia, and other physical properties. Examples of physically derived motion are a rigid object falling down stairs and a flag swinging in wind.

An animation system can use techniques developed in one or several problem domains, to achieve the best control effect for an animation. A system can use dynamic control techniques to produce physically realistic motion, or kinematic control techniques to get good feedback rate, or highly interactive 3-D control devices to directly specify a motion. The combination of techniques used in a system may vary from one system to another. However, the ultimate goal of a system is to offer the user an intuitive, creative, and structured control environment. This environment offers good evaluation criteria for using techniques. That is also why we include typical systems in the discussion of animation research.

Our discussion of animation research in each problem domain covers the following aspects: the basic theories on problem solving, research progress, important contributions, and typical systems. Since the discussion clearly outlines the past research experience in each problem domain, further research on similar problems can be better planned. Also, the discussion encourages the exchanging of ideas between different domains.

Three problem domains have been identified by this research: control approach, control facility, and control hierarchy. The domain of control approach deals with the techniques used for controlling a motion. The domain of control facility deals with the tools used for specifying the control of an animation. The domain of control hierarchy deals with the use of hierarchies for controlling complex motions. In each of the three domains, two major groups emerged as alternative researches are the kinematics and physically derived motion groups in control approach, interactive tools and programming languages groups in control facility, and high-level abstraction and multi-level interfaces groups in control hierarchy.

Kinematics and physically derived motion are the two major control approaches used in animation, where kinematics studies the conventional motion control using geometrical transformations and physically derived motion (also called dynamics) studies the use of forces, torques, and other physical properties in an animation. Interactive tools and programming languages appear as the two major approaches dealing with the issue of how to specify the control of a motion. For the control hierarchies, high-level abstraction allows convenient and automatic motion generation but lacks control over the details of the motion. On the other hand, multi-level interfaces provide control at levels of progressively more detail.

2.2. Kinematics vs. Physically Derived Motion

2.2.1. Kinematics

The Approach

Kinematics is the traditional animation approach in which motion is described by geometrical transformations, such as rotation, scaling, and translation. No knowledge of the physical properties of the object are used in motion control. This approach has been widely used in the control of keyframe animation. As the extremes of a motion or critical junctures in a motion sequence are explicitly specified by keyframes, using geometrical transformations, in-between frames are generated by various interpolation methods such as linear interpolation, parabolic interpolation, or cubic spline interpolation.

Linear interpolation is fairly straightforward and well suited for transforming the shape of an object. It may, however, produce some undesirable side effects which give the animation a mechanical look. The main drawback of linear interpolation is the lack of smoothness in the resulting motion. Alternatively, parabolic interpolation maintains the smoothness of a motion path by its continuity in velocity, and cubic spline interpolation generates a motion with continuity in acceleration.

The control of kinematics can be conducted either by forward kinematics or inverse kinematics. Forward kinematics is the positioning of a point by traversing the transformation matrices from some root node outward to the leaves of a modeling tree. Inverse kinematics determines the rotation angles or the translation values from the end effectors located at a particular point in space. As kinematics controls a motion based on the computation of geometrical transformations, the motion involved in a complex environment can be difficult to produce using this technique.

Research Progress

In the early 1960s, the temporal behavior of interpolated points on a path was first noticed by animation researchers. The P-curve technique was introduced by Baecker [Baecker69] in his GENESYS system, where a P-curve was used to define both the trajectory of the point and its location in time. This technique

was later extended to three dimensions by Csuri [Csuri75]. Further in this direction, the technique of using moving point constraints between keyframes was developed by Reeves [Reeves81]. This technique allows the specification of multiple paths and the speed of interpolation, which is used to generate a patch control network. One example with three keyframes (k1, k2 and k3) and three moving points (m1, m2 and m3) which produces an interpolated path over equal time intervals is shown in Figure 2.1.

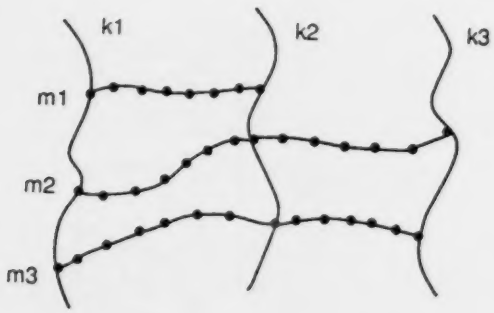


Figure 2.1 In-betweening Using Moving Point Constraints

The use of kinematic positioning coupled with constraint specifications has been suggested by Badler [Badler86] as a promising solution to complex animation tasks. By definition, a constraint includes spatial regions, orientation zones, and time expressions. Multiple constraints on a body position or orientation may be specified and optimally controlled with a constraint satisfaction system. Three-dimensional input devices have been used for positioning and manipulating three-dimensional objects and for visually establishing multiple constraints and motion goals.

The difficulty in applying the keyframe approach to three-dimensional animation is due to the description of three-dimensional keyframes. Interactive and programming descriptions of three-dimensional keyframes are not natural, as the descriptions are based on the projected keyframes in two dimensions. In order to obtain a realistic three-dimensional effect, the description may need to be refined many times. This repeated cycle can be very costly and time consuming when the described keyframe becomes complex.

Towards this problem, the idea of "keyed" parameters, which in turn control the positioning of objects in the keyframes, has been developed. Once an appropriately parameterized model is created, the parameters can be keyed and interpolated to produce an animation. One well-known example of parameterized model is the facial animation created by Parke [Parke82].

System Application

GENESYS [Baecker69] is an early picture-driven animation system developed at MIT. In this system, a motion can be described by using a set of predefined animation commands, and it can be adjusted in terms of the dynamic behavior of the parameters on each P-curve path. For instance, the motion from one corner of a room to another can be controlled by the parameters $x(t)$ and $y(t)$ on each P-curve. The system also provides the rhythm descriptions to vary the display time or intervals between frames.

SCANIMATE [Honey71b] is an analog animation system developed by Computer Image Corporation. This system allows the signals produced by a video synthesizer to be modified. An image can be zoomed, shrunk, and rotated, and the color and intensity of the image can be easily modified. The system has been used for many commercials and films, including "2001: A Space Odyssey" and "Yellow Submarine". The CAESAR system [Honey71a] is similar to SCANIMATE, but with more operational power. This system allows a figure to be segmented into several parts, each of which can be separately controlled and displayed on a separate portion of a TV screen. The inbetween frames can also be computed by the system. This system has been used by the ABC, NBC, and CBS TV networks.

GRASS [DeFanti76], designed by the Computer Graphics Research Group at Ohio State University, is a user-oriented real-time animation system. The system is described as "habitable" as it could be used by a computer novice. The commands for changing intensity and moving a picture along a path are similar to Baecker's P-curve technique. The commands which change the vector list, such as translation, rotation, and scaling, are performed by the hardware. Macros and conditional execution are also provided in GRASS.

BBOP, developed at the New York Institute of Technology [Stern83], is designed to animate

hierarchically

articulated 3-d models by interpolating keyframe poses established during interactive sessions. The position of the model at any one instant is determined solely by nested transformation matrices. EM [Hanrahan87, Hanrahan87] is a system developed as an enhancement to BBOP to bring a form of programmed control to the keyframe approach. EM uses a geometric modeling language which allows parametric control over the transformations at each joint in the model and over the geometry of the individual body parts.

TWIXT [Gomez86], a three-dimensional event driven animation system, is designed by the Computer Graphics Research Group of Ohio State University. In contrast to traditional keyframe interpolation schemes, TWIXT constructs frames to capture the display parameters of the objects onto the tracks of action. On these tracks events are defined whenever something happens. The values between events are interpolated based on frame numbers. The union of activity on all tracks forms a frame instance.

2.2.2. Physically Derived Motion

The Approach

Physically derived motion applies physical laws to derive the object's movement, instead of positioning the object by geometrical transformations. In this approach, a motion is calculated in terms of the object's mass and inertia, the applied force and torque, and other physical effects of the environment. As a result, the motion produced is physically more accurate, and appears more attractive and natural. There are many types of motion, such as falling or reacting to collisions, which can result automatically from the dynamics environment. In animation research, it has been a goal to establish the control techniques for dynamics and use dynamics to provide a minimal user interface to the highly complex motion. Besides, the dynamics approach is generally considered a useful tool in the fields of robotics and biomechanics.

Physically derived motion, also called dynamic control, takes into account a body's mass and inertia

as well as the various forces acting on the body. The dynamics equations of motion are used to relate the acceleration of the mass (object) to the forces and/or torques acting upon it. A force produces translational motion and a torque produces revolute motion. The well known Newtonian equation of motion, $F = m*a$, is used to produce the motion of a particle. In this equation, F is the force vector acting on the point mass, m, and a is the acceleration the mass experiences. Given the acceleration, the velocity and position along the path of motion can be found. A torque is produced when a force acts at a point on the body other than the center of mass. The basic equation for computing torque has the form $T = p*f$, where $p(x,y,z)$ is the point being acted on and $f(fx,fy,fz)$ is the force applied to it. Similar to a force, a torque can be represented as a 3-D vector in space. Moreover, other types of forces, such as gravity, spring, and damper, can also be modeled and integrated into the dynamic environment when they are necessary.

Research Progress

A wide variety of dynamics formulas have been discovered since 1500. One example is Newton's three laws of motion. These laws explain why objects move and the relationships which exist between force and motion. The methods used for integrating individual forces in 3-D vector space are well defined in physics. In computer animation, various forces and torques acting on and in the object's body can be abstracted to a few control types. For instance, the gravitational force can be calculated automatically. Interactions with the ground, other collisions, and joint limits can be modeled as springs and dampers. Such internally controlled motion as muscles in animals or motors in robots can be interactively specified using an interactive interface. In Figure 2.2, an example of the forces and torques acting on a body is presented.

The dynamics equations for articulated bodies including humans and animals have posed a research challenge for computer animation. An articulated body is modeled with rigid segments connected together at joints capable of less than 6 degrees of freedom. Because of the interaction between connected segments, the dynamics equations are coupled and must be solved as a system of equations, one equation for each degree of freedom. Numerous formulations of the dynamics equation for rigid bodies have been defined. Although derived from different methodologies, all the equations produce the same results. The most

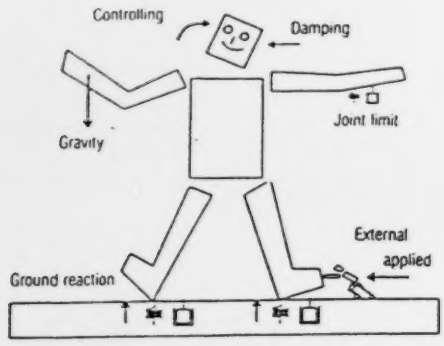


Figure 2.2 Forces and Torques Acting on the Body

significant equations include the Euler equations [Wells69], the Gibbs-Appell formulation [Horowitz83, Pars79, Wilhelms5a], the Armstrong recursive formulation [Armstrong79, Armstrong85], and the Featherstone recursive formulation [Featherstone83].

The Euler equations are one of the more intuitive formulations. These equations are defined by three translational equations and three rotational equations. It is simple to solve these equations if the accelerations are given and the forces and torques are desired. But, the equations do not properly deal with constraints at the joints. The Gibbs-Appell equations are a non-recursive form that has $O(n^4)$ time complexity for n degrees of freedom. The equations express the generalized force at each degree of freedom as a function of the mass distribution, acceleration, and velocity of all segments distal to this degree of freedom. Thus, the method allows considerable flexibility in designing joints. The Armstrong recursive formula can be thought of as an extension of the Euler equations with multiple segments. The method is built on tree structures and is suitable for certain types of joints. The complexity of the method is linear in the number of joints. The Featherstone method is a recursive linear dynamics formulation, and is flexible in the types of joints. For the control of articulated figures, the recursive method can compute the motion more efficiently than the independent equation method if the figure contains more than nine joint levels.

As dynamics offers hope for more realistic, natural, and automatic motion control, the approach has

also been introduced into kinematic models as a supplemental tool. Users may specify the motion of a limited subset of body joints and have the dynamics calculate appropriate motion for the rest. Inverse dynamics is a common technique used for "goal directed motion". The technique computes a motion starting from the desired end place. The placement of an object's end effect can be dealt with by pushing and pulling it with external forces, which can be further utilized to generate the motion in between. Position constraints can also be simulated by using appropriate forces holding the body in position. The model of a mass-spring control system has been used to control the motion of legless figures such as snakes and worms.

The use of dynamic control for the problem of collision response between rigid objects is discussed by Moore and Wilhelms [Moore88]. In this technique, a collision is treated as a kinematic problem in terms of the relative positions of objects in the environment. The response of arbitrary bodies after collision in the environment is modeled using springs, and an analytical response algorithm for articulated rigid bodies is also applied to conserve the linear and angular momentum of linked structures.

A general control model used for the three dimensional dynamic process of arbitrary rigid objects has been proposed by Hahn [Hahn88]. This model takes into account various physical qualities such as elasticity, friction, mass, and moment of inertia to produce the dynamic interactions of rolling and sliding contacts. Another technique used for the dynamic control of collision between rigid bodies [Baraff89] starts from the problem of resting contact. The forces between systems of rigid bodies, either in motion or stationary, with no-colliding contact are analytically formulated, which can then be modified to simulate the motion of colliding bodies.

Spacetime constraints [Witkin88] is a technique that combines both the what and how requirements of a motion into the system of dynamic control equations. A motion can be described not only by what task is required, such as "jump from here to there", but by how the task should be performed, "jump hard or little". These requirements are specified by coupling the constraint functions representing forces and positions over time to the equations of the object's motion. The solution to this problem is to control the motion which can satisfy the "what" constraints with the "how" criteria optimized.

Another notable research direction in dynamic control is the modeling and animation of elastically deformable materials, such as rubber, cloth, paper, and flexible metals. This technique employs elasticity theory to construct differential equations that represent the shape and motion of deformable materials when they are subjected to applied forces, constraints and interactions with other objects. The models are active as they are physically based; the descriptions of model shape and motion are unified to yield realistic dynamics as well as realistic statics in a natural way.

System Application

Near-real-time control of human figure models has been produced using an interactive system at the University of Alberta [Armstrong85]. The system has two main components: front end and dynamic analysis. The front end component is responsible for displaying the human figure model and interacting with the user. The dynamic analysis component manages a collection of motion processes. Efficient algorithms for solving the equations of figure's motion have been used in a distributed computing environment of multiple processors.

The Deva animation system [Wilhelms87], developed at the Berkeley Computer Graphics Laboratory of the University of California, is an experimental system for simulating the motion of articulated bodies using the Gibbs-Appell formulation. In this system, the forces and torques are controlled in one of the five states: direct dynamic control, relaxation, frozen, oriented, or hybrid k-D control. To ease the control complexity, the graphical editor Virya is used with the system. With Virya, the user can design and store controlling functions for each degree of freedom of the body, as well as the state of each degree of freedom over time.

The Manikin system [Forsey88] is used to explore dynamic manipulation of articulated bodies at the University of Waterloo. This system uses dynamic analysis to address the traditional problems of inverse kinematics, directed movement, and kinematic constraints. The system shows that it is possible to have multiple forces, torques, goals, and constraints applied simultaneously. The key positions of articulated

bodies are dynamically manipulated, and later interpolated by a spline function. The forces and torques applied in a dynamic control are automatically derived by the system.

Other animation systems use dynamic control in addition to their kinematic models. The PODA animation system utilizes simple dynamics to simulate the forces applied by the legs. Thus, the acceleration of a figure's body is manipulated by applying forces to produce coherent dynamic realism. The famous work of mechanical ants and robots by the New York Institute of Technology also includes some dynamic analysis in the control. These systems suggest the possible integration of dynamic control with kinematic models, when only the appearance of realism is required but physical reality is not.

2.3. Interactive Tools vs. Language Programming

2.3.1. Interactive Tools

The Approach

Interactive techniques have attracted considerable attention in animation research due to their flexible control capabilities. Basically, interactive control refers to motion control techniques that allow the user to design motions in realtime while watching the animation develop on the graphics screen. For example, the "keyed" parameters of a model can be continuously modified by connecting their values to numeric input devices. The values of parameters can be stored in a hierarchically structured database. When reading from the database, parameter values between keyframes are automatically interpolated to produce the animation sequence.

Research Progress

In using interactive control, three primary tools for setting up interaction modes are used. These are systems for relating input devices to parameters, menu facilities, and methods for programming based on keyboard and function keys. In order to control parameters with input devices, inputs are modeled as variables whose numeric values are functions of physical devices. Statements connecting parameters to input

devices are entered from the keyboard or read from a file. The assignment of inputs to devices can be designed to take advantage of characteristics of input devices, such as using the tablet to control the motion of a whole body, and the joystick for an arm. Moreover, the construction of constrained inputs can be produced by connecting functions. A group of parameters can be controlled by a single device, or a single parameter's value can be produced by a function combining the input from several devices.

The menu system allows users to create their own system of menus to interface with the command interpreter. The items in a menu can be displayed on the screen and selected using a tablet. A set of menus can also be set up so the user can organize a network or hierarchy of menus. The user may either use a set of default menus, or create and organize a set of individualized menus on a per-model, or per-user basis.

The alias facility of a system allows users to assign commands to one key or a sequence of keys. For instance, if a command contains a particularly long specification, it can have an alias that is either an abbreviation, or a single key on the keyboard. The command interpreter responds to the abbreviation or key exactly as if the full command had been given. More generally, partial commands and strings can also be aliased. Sets of aliases can be saved and used for different control models.

Interactive techniques have generally been used in three control processes, which are keyframing, path specification, and control functions. In keyframing, the user specifies keyframes, a sequence of positions and the times when they occur, and the computer interpolates between these positions to produce the animation. Path specification involves interactively designating a coherent path over time. Often this path is defined by using a tablet to pick positions in a displayed world. The description of the path represents the entire motion of a particular object. The use of control functions defines the motion for each degree of freedom as a function of time versus position. Again, the functions are generally developed by designating control points which define curves.

A number of experiments employing three-dimensional interactive devices have been conducted. One of them that uses a field sequential stereoscopic display to facilitate the task of 3-D object pointing [Takemura88] has been developed at ATR Communication Systems Research Laboratories. Three performance

aspects of pointing interaction were measured. First, human performance in adjusting random dot stereogram depth is measured to determine the possibility of pointing at a 3-D object. Second, user performance in pointing at 3-D objects using a mouse as an input device is measured. Third, user performance using a 3-D magnetic tracking input device is tested. These experiments show that the task of pointing at a 3-D object on a field sequential stereoscopic display can be performed with relatively high accuracy, and the 3-D magnetic tracking device has better performance than a 2-D input device like a mouse in terms of the task completion time and error rate.

Another notable research result using a six-dimensional sensor for object placement was presented by Ware [Ware88]. As the problem of placement is viewed as a six-dimensional operation, three for position and three for orientation, a six-dimensional sensor called a bat is configured using a one button six-dimensional mouse. Both the ability to perceive the spatial relationship of objects in the three-dimensional environment and possible user interactions to replace the objects in the environment using the bat are tested on an IRIS workstation. A variety of interaction modes, such as translate x and all translations by x, y, and z, can be selected from a system menu. Even though the research did not directly address an animation problem, the fundamental achievement of the research is to show the effective use of an interactive device in the three dimensions.

The concept of virtual reality, which simulates the natural action in the real space, has been recently introduced, and used in the research of space touring and interactive manipulation within a three-dimensional environment. The research group at University of Alberta [Green90] has reported some interesting results on a virtual reality control environment using several interactive devices in one system. The system can interact with the user by both his/her eye and hand movements. During the interaction, the user can view a three-dimensional image wearing a "EyePhone", and touch or grab an object displayed in the "EyePhone" with his hand's movement, where the movement is detected by the "DataGlove" worn on the hand. Thus, the system provides the user with a virtual space in which the task of interactive control can be performed by the joint efforts of a user's eye and hand movements.

System Application

TWEEN [Catmull79], designed at NYIT's Computer Graphics Laboratory, provides a means of generating and manipulating digital forms of character images. At any time, the user can review the in-betweens and modify them with an electronic pen. The resulting changes to other in-betweens are recalculated by the system.

PODA [Girard85], designed at Ohio State University, allows animators to control the complex relationships between the motion of the body of a figure and the coordination of its legs. An interactive menu-driven interface is used for both the incremental construction and behavioral control of animals having any number of legs composed of any number of joints. An example of a 14 legged insect shown in its walking cycle in PODA is given in Figure 2.3.

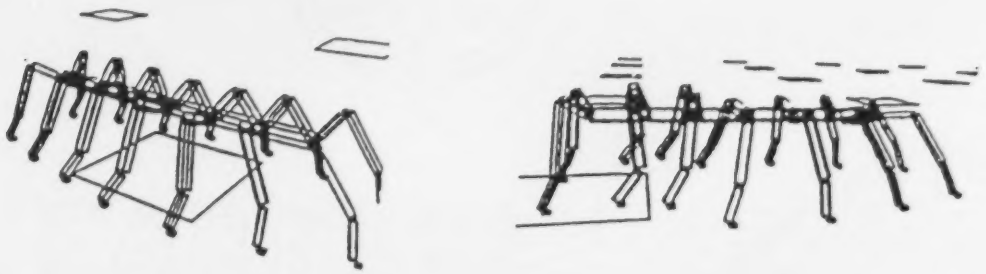


Figure 2.3 A 14 Legged Insect in Its Walking Cycle

The 3SPACE Digitizer System [Badler86], implemented at the University of Pennsylvania, makes use of a six degree-of-freedom input device for manipulating and positioning three-dimensional objects. With this device, the positioning of an articulated figure is handled by visually establishing multiple goals, and then letting a straightforward tree-traversal algorithm achieve simultaneous satisfaction of all constraints. An example of setting a goal for the lower torso and a goal for the hand of a figure is shown in Figure 2.4.

MUTAN (Multiple Track Animator) [Fortin87], designed at the University of Montreal, is an interac-

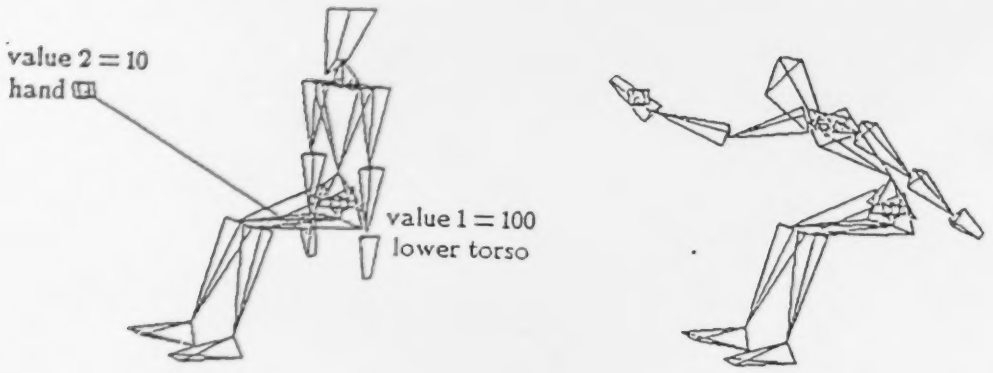


Figure 2.4 Interactive Setting of the Goals of the Lower Torso and the Hand

tive system for independently animating 3-D graphical objects. All animation constraints for a graphical object are recorded on separate tracks. A set of commands which allow the animator to manipulate marks, tracks and frames are defined. The system also includes the 3D HORIZON graphics editor and a 3D digitizing program.

2.3.2. Programming Language

The Approach

Animation languages provide a programming means for specifying and controlling animation. The object modeling, temporal relationship of parts, and motion variations can be explicitly described by a stream of textual instructions in a programming language. Using an animation language gives the animator complete control over the process. The motion concepts and processes can be expressed in terms of abstract data types and procedural programming. Once a program is created, the rest of the process of producing the animation is completely automatic. The programming approach is suitable for algorithmic control or when the movement simulates a physical process. Certain sophisticated motions and some special motion effects can be animated with a slower programmed approach. One major disadvantage in using programming, however, is its feedback cycle. The animator may not see the resulting motion until the program is complete

and the full animation is rendered.

Research Progress

There are three subbranches which have emerged in the development of animation languages. These are subroutine libraries, preprocessors, and complete languages. Subroutine libraries are used to supply graphical functions that are added to a preexisting high-level language. The library can be simply linked to the language at execution time. Examples of subroutine libraries include the ACM Core system and the Graphics Kernel System (GKS). These graphics packages support two and three dimensional transformations, perspective, drawing primitives, and control structures. A subroutine package can be both language and device independent. The cost of using a subroutine package is fairly low. But the calling interface to subroutines can be difficult to use.

A graphics preprocessor is an extension to a compiler that permits the syntax of an existing language to be augmented by new (graphics) commands. New graphical features are recognized and incorporated into the language. The preprocessor program works prior to the interpreter or compiler, and its output, which combines native language commands with expanded commands, is passed to the high-level language compiler and compiled as usual. Through the use of this technique, most high-level structures of programming, such as loops and condition flow, are supported by the host language. From the user's viewpoint, a new graphics language that fully incorporates an existing high-level language as well as graphics commands is created. The new graphics language is executable if the graphics commands are precompiled to the host language environment. Consequently, the technique has been widely used in the design of graphics languages.

A complete programming language with original graphics syntax and semantics is yet another approach to developing, manipulating, and displaying visual images. In this approach the expense of a preprocessor is avoided, but considerable effort is required to produce a complete programming language. Also, a new compiler is required for the language. In practice, few graphics languages have been designed

using this technique.

Language Application

A number of object-oriented and actor languages have been implemented to date. LOGO [Papert70], developed at M.I.T., is a programming language primarily for children. Its most important concept is turtle geometry which is based on polar-coordinate graphics.

ASAS [Reynolds78, Reynolds82], designed at the Architecture Machine Group, is an extension of the Lisp programming environment. This language includes geometric objects, operators, parallel control structures and other features to make it useful for computer graphics applications. The operators are applied to the objects under the control of modular programming structures. These structures, called actors, allow parallelism, independence, and optionally, synchronization. Also, the extensibility of ASAS makes it grow with each new application.

CINEMIRA [Thalmann84] is a high-level, three-dimensional animation language based on data abstraction. The language is an extension of the high-level Pascal language. It allows the animator to write structured scripts by defining animated basic types, actor types and camera types. For instance, a tree model can be described by a 3-D graphical type as follows:

```
Type TREE = figure( Var BRANCHES:  TEXT;
                    NBRANCHES: INTEGER;
                    POSITION:  VECTOR;
                    HEIGHT,LENGTH: REAL);
```

where BRANCHES is a file of kinds of branches, NBRANCHES is the number of branches, POSITION is the position of the trunk, HEIGHT is the height of the trunk, and LENGTH is the length of the branches. Another example which shows the motion sequence of a bird could be specified as:

The class hierarchy of a modeling and display system [Grant86], designed at the University of North Carolina, has been applied to the problems of constructing extremely complex geometric objects. The system is defined in terms of the class facility of the C++ language. With the class hierarchy, the common elements of geometric procedures and methods are shared by as many classes as possible. An example of the

```
Procedure DRAWBIRD( FRAME: INTEGER);
  Var FIRSTBIRD: BIRD;
  begin
     create FIRSTBIRD(FRAME,RIGHTBODY,RIGHTWING,C,D);
     TRANSLATION(FIRSTBIRD,<<0,0,FRAME*BIRDSTEP>>,FIRSTBIRD);
     draw FIRSTBIRD;
     delete FIRSTBIRD;
  end;
```

class hierarchy of the system is shown in Figure 2.5. Rather than develop a uniform primitive representation, the system accepts a diversity of geometry by building a framework which combines dissimilar models in an orderly manner.

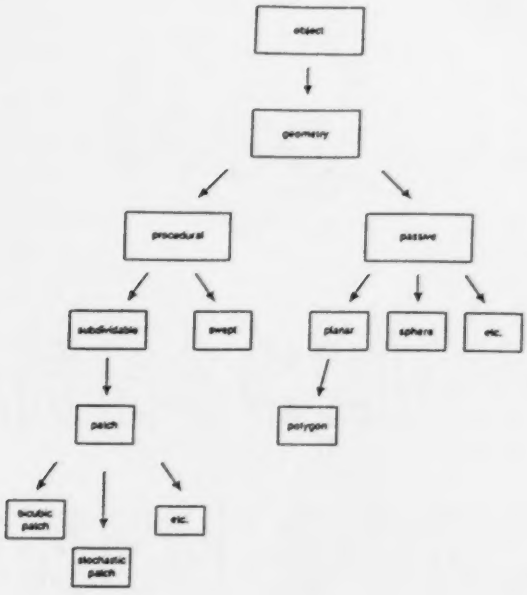


Figure 2.5 The Class Hierarchy in the Modeling Portion

As it turns out, the production of three-dimensional animation using a graphical programming language is still time-consuming. The quality of the animation is closely tied with the programmer's knowledge and skill. In addition, the description of irregular or stochastic processes is not natural if expressed by language statements. One alternative to these problems is to provide a programming interface

to an interactive animation system. Examples can be drawn from current systems. EM, for instance, brings a form of programmed control to the keyframe animation system BBOP. MIRANIM [Magnenat-Thalmann85] is composed of an animator-oriented system called AMMEDIT and an animation sub-language called CINEMIRA-2. As a result, more control facilities and less programming knowledge for complex motions are both offered by the system.

An extended graphics language is influenced by the programming language it is based upon, and an interactive system strongly relies on hardware facilities for controlling interaction in realtime animation. In between these two approaches, scripting has been applied in an effort to provide a method of animation that is more powerful than the interactive approach and not as difficult to work with as a full blown programming language. A script language is usually easier to learn and use than a general-purpose programming language and thus easier for people who are not experienced programmers.

2.4. High-level Abstraction vs. Multi-level Interfaces

2.4.1. High-level Abstraction

The Approach

One of the current research trends in animation is to provide the user with higher level control. By providing a means where motion can be automatically controlled, much of the burden of generating detailed motion descriptions is shifted from the animator to the animation software. The animator is able to communicate with the animation system in what amounts to a high-level interface for automatic motion synthesis. For instance, a motion can be described as "walk towards door" or "open window", leaving the system to find the appropriate low-level motion description. These high level control ideas have also been used in the other fields, such as robotics and artificial intelligence.

The Development and Examples

A natural language interface and a knowledge base which encodes an understanding of some simple English verbs for motion has been proposed at the University of Pennsylvania [Badler89, Kalita89]. If it is possible to input natural language to an animation system, we can expect that particular actions can be modeled by adverbial modifications of the closest motion verb. It can also be expected that this system will use English verbs to characterize more complex activities and, in particular, incorporate the mechanical characteristics of the tasks in the lexical meanings of action verbs.

Facial animation is another example of high-level abstraction. A number of facial animation systems have been created. The common characteristics of these systems are the high-level abstraction of the facial parameter model. For instance, in Parke's system [Parke82], the expressions and features of the face are controlled by two classes of parameters: expression and conformation. Expression parameters are related to the eyes and the mouth, and include such things as pupil dilation, eyelid opening, direction of vision, jaw rotation, and width of the mouth. Conformation parameters include the color of the skin, the color of the eyes, the neck dimensions, nose characteristics and so on. Using this model, each emotional expression, like "shout" or "giggle", can be produced with a natural expression.

Various techniques have been developed to convey high-level to low-level interpretation. These include parameterization, finite state machines, command libraries, and hierarchies. Parametric motion control involves designating parameters whose values define the configuration or motion of the objects modeled [Hanrahan87, Parke82]. Deciding the best way to structure the parameters that represent the desired motion is one problem in using this technique. Finite state machines are appropriate for describing and controlling repetitive or coordinated motion [Tomovic67, Zeltzer82]. The state changes rely only on the current configuration of the model and the passage of time. Command libraries provide a means of storing low-level motion descriptions under high-level command names. However, combining commands may send different directions to the same joints and produce nonsensical motion. Use of a hierarchical structure which parses high-level commands through levels of progressively more detailed control hierarchy may make combining commands less problematic and also reduce the number of commands that need to be

stored. The possibility of combining these techniques into one system is very promising. One example is Zeltzer's skeleton animation system [Zeltzer81-Zeltzer83] which combines the use of command libraries and a hierarchical interpreter with finite state machines.

2.4.2. Multi-level Interfaces

The Approach

The high-level abstraction of a motion is supported by a number of low-level processes. The high-level abstraction allows a natural, goal-directed communication between animators and animation software. But, it lacks the control flexibility of the low-level motion processes. When the motion is complex, there are many degrees of freedom between the model's parts and between the model and the environment, which need to be considered. The control properties and relations in motion need to be abstracted into levels. The capability to select and manipulate the low-level processes needs to be supported.

A human body possesses some 200 degrees of freedom controlled by about 400 different muscles. In terms of a motion goal, the activity at each joint must be defined over the time interval of the motion. Without a multiple level control scheme, the motion will be rigid and less flexible. For instance, how can the "walk" style be varied in terms of the speed of pace, the swing style of the arms, and the walking distance between paces? The variation of "walk" style depends upon the walker's physical state at the moment (tired or full of energy) and the physical structure of the body (short or fat). These physical states and geometrical conditions can be controlled through the levels of a structured parameter model.

In order to control a complex motion, a large problem with many degrees of freedom needs to be decomposed into a hierarchically coordinated set of smaller subproblems, each with only a few degrees of freedom. Commands to the top level of such a system invoke the set of low-level subsystems necessary to perform a particular motion sequence. Also, the variations of motion can be manipulated at different levels of detail in the hierarchy. The idea of problem decomposition through hierarchical levels has been used as a powerful organization principle. This organization promises a system which will be general, relatively easy

to understand and control, and extensible.

One example of using a control hierarchy in animation is the PODA animation system [Girard85]. The system uses a strategy in which the figure's motion may be designed and manipulated at different levels of control. At the lowest level the animator may define and adjust the character of the movement of the legs and feet. At a higher level the animator may direct the coordination of the legs and control the overall motion dynamics and path of the body. More examples of using the control hierarchy of animation will be presented in Chapter 5, in comparison to the control hierarchy proposed by this thesis research.

2.5. Approaches Using the Animation Techniques

Several animation approaches have applied the techniques developed in animation research in various areas. These are described as the algorithmic approach, stochastic approach, learning approach, stimulus-response approach, behavior rule approach, predefined environment approach, robotics research approach, and parallel computing approach. Among them, the algorithmic approach refers to motion developed through the use of a particular algorithm, possibly some mathematical expressions. The stochastic approach is based on the use of random perturbations applied during modeling and motion, which are usually defined by other methods. Examples using this approach are fractal mountains [Fournier82], particle systems [Reeves83], and grass blowing in wind [Reeves85]. Learning refers to changing the motion depending upon experience. By learning, objects can be controlled, as real creatures are, with the ability to think. Thus, motion can be much more interesting and attractive, rather than mechanical-like.

The stimulus-response approach suggests that environmental interactions be taken into account during motion generation. The motion of each object is determined not only by its own internal algorithm, but by the behavior of other objects in the environment. Thus, the motion is usually triggered by the output from a neural control network which takes input from the object's sensors. Examples of this research are the works of Braitenberg [Braitenberg84], Travers [Travers88], and Wilhelms [Wilhelms89]. Similar to the stimulus-response approach, the behavior rule approach addresses the motion depending on other objects

and the environment sensed by the moving object. But, rather than using the neural network, this approach uses a set of behavior rules which determines when an action should be called. Examples of this approach are Petworld [Coderre88] and Reynold's flock-like behavior [Reynolds87]. The predefined environment approach suggests another solution to the behavior control problem in an environment. This approach makes the assumption that the environment is previously known. One optimal motion path is selected from all other possible solutions computed from the environment. Animation research related to this approach are the works of Ridsdale [Ridsdale88], Hahn [Hahn88], and Moore and Wilhelms [Moore88].

Similar researches have been done in the related fields of motion control in computer animation and mechanical simulation in robotics. Examples include forward and inverse kinematics, motion trajectories, dynamics, collision detection, and path planning. The current state of the art is similar in both fields. For instance, some robotics systems use "teaching by example" for control, where a robot arm is positioned at certain points in a trajectory. The system remembers these positions, and generates a smooth motion path through them. This method follows the same principle as keyframing or path specification in computer animation. Other robotics systems control the robot by a series of instructions, which corresponds to the research of animation languages.

Another animation approach is to use the power of parallel processing to reduce the computational load of an animation. This has the potential of providing a real-time motion control environment for many types of interactive animation. One example is the FrameWorks animation system [Green87] that handles the distribution of the motion computations over a network of workstations. This system utilizes the inter-frame dependencies over the network and supports the task level to parallelize an animation program over the network. Towards the similar motivation, a survey on the current parallel architectures and algorithms for graphics rendering is presented by Crow [Crow88].

Chapter 3

Problems Review and Research Outline

This chapter discusses the details of special control issues involved in the problem domain of scene animation. Research motivations towards the identified scene problems are outlined. Prior to the proposal of a new approach, previous approaches motivated in the similar research direction are reviewed and evaluated. The basic ideas in our proposed relation control approach are briefly explained. And an example using the relation control ideas is illustrated.

3.1. Review of Problems

The problem of motion specification for a single object has been studied for decades. However, the problem of scene specification has not been thoroughly studied in current animation research. There are many special control issues which are not involved in the motion of a single object, but are important for motion in a scene environment. In the following subsections, these issues are discussed under three headings: large problem size, implicit behavior structure, and indirect environmental control.

3.1.1. Problem Size -- Number of Degrees of Freedom

Animating a single object's motion can be difficult due to the large number of degrees of freedom in the object's model. Examples of such models are trees with many branches and human figures with a large number of body segments. An object modeled by a large number of control degrees of freedom (object parts that can move) implies a large control problem. The task of animating an object with a large number of degrees of freedom is more complex than animating one with a simple model, such as a ball or box. To further illustrate this point, consider a human figure model as an example. It can have over 200 degrees of freedom when a fully defined articulated figure model is used. Imagine the case of animating such a realistic figure's motion, which can require the specification of tens of thousands of values for few seconds of the figure's movements.

A large number of degrees of freedom in an object's model presents the first complexity layer of an animation. This layer can not simply be removed or transferred to another layer with less control complexity, even though inverse control or goal-directed task animation in some sense can reduce some of the difficulties. In general, the control-degree layer must be dealt with by an animator, in some clever and efficient way. Above this layer, there are other layers which create additional control difficulties in animation. One of them is to guess the "correct" control values for a motion. Determining the "correct" result for an ill-defined model introduces additional control degrees for an animation, especially when the model contains a large number of controls. It may be possible to determine the "correct" control values for a motion, but these values have little to do with a slightly different motion task or goal. Consider the human figure example again. Since the figure has a number of coupled subparts, the movement of one subpart will be influenced by the motion of the other connected subparts, directly or indirectly. The influence of these subparts can result in a very complex situation. Imagine the cost and time required to guess the "correct" control values for the figure's motion, if the figure model is not well-defined and understood.

It is not impossible to guess an object's motion once, but in general the trial and error approach can be very expensive and time consuming. Current animation research has addressed this problem for a single object's motion, using well-defined object models. But, the research has not addressed the problem of scene motion, the motion in the dynamic environment with both static and dynamic objects, environment boundaries, and unpredictable scene events. A scene motion is an extension of a single object's motion. In this domain, an object's motion is not isolated to the object itself, but dynamically influenced by the environment in a very complex control situation.

The additional control degrees introduced by an environment can be seen from the example of two moving objects. When two objects are moving together, their motions are influenced by each other. One natural influence is the avoidance behavior between the two objects. While avoiding each other, one object might change course, use a different speed, or perform different actions. One moving object can show interest in the other, or have a dislike feeling to the other, while the other follows, copies, and disturbs the

first one's motion. During the animation, one object can also play a leading role and encourage or assist the other object. As a result, the leading role may be discouraged, ignored, or rejected by the other object, or influenced in another way. To effectively address the additional control complexity between the two moving objects, the additional degrees introduced by the other object should be modeled.

Here, we use the term "connection" to describe the influence of the environment on a motion. This connection is different from a physical connection in an object's model, because the connection does not exist physically, but virtually in a condition that each environmental influence on the motion must or may satisfy. For the two moving objects example, if one of them stretches his arms, the other may respond to the stretching in some way. The other object may need to avoid the stretched arms or perform a similar reaction. The important question here is how the motion in a two-object environment can be controlled to automatically lead to a natural and realistic animation.

When the two-object environment is extended to a more general environment with a boundary, obstacles, other static and dynamic objects, and scene events, the motion of an object is further constrained by the environment. In this case, the motion is not just affected by the other object as in the two-object environment, but by the boundary, obstacles, other static and dynamic objects, and scene events in general. Each of them may contribute to the modeling of an object's motion. Avoiding the possible collisions with other objects in the environment is the first consideration. This consideration will vary an object's motion whenever a potential collision could occur. We refer to this as the environmental collision connection, which is implicitly connected from the moving object to the other objects in the environment.

Besides the collision connection between a moving object and the environment, many other environmental connections can be modeled. One example is the "connection" between a moving object and the environment boundary. The object may be interested in one piece of the environment boundary but bored with the other pieces. This interest may lead to a motion approaching and following along the special piece, and a bored feeling may turn the moving object away from the other pieces. From the two-object environment to the general environment, the "connection" between the two objects is extended to every pair of

moving objects and the other object in the environment, which forms a large control basis for the motion.

If the "connections" from a moving object to the environment are not explicitly modeled, the task of animating the motion must rely on guessing how a natural scene motion should be. Even when the problem we are discussing is shifted from a single object's domain to a scene domain, the essential problem is the same -- to animate a natural motion which is defined by a large control basis. So we can compare the scene motion case to a single object's motion by the pain of guessing the "correct" control values. As indicated above, it can be very costly to guess the "correct" values in a human figure model. A similar situation occurs when the motion is involved in the dynamic environment.

It may be argued that it is always possible to animate a scene motion according to one predefined sequence. Thus, the scene problem can be solved in the same way as for a single object, where the scene motion is predefined by a sequence of actions. In this case, it doesn't seem necessary to study the general control issue of scene motion, such as how a motion can be influenced by the environment. There are two answers for this argument. One is the additional difficulty in the dynamic environment of guessing the "correct" control values for the motion. Blind guessing does not reduce or solve the additional degrees introduced by the environment. Instead, these environmental degrees grow with the number of objects involved, and the complexity of the motion to be modeled. The other answer is the wide range of environmental behavior applications. In an environment, there are a large number of behaviors that can be explored. If we do not understand how a motion is generally "connected" to the environment, each behavior application requires a predefined sequence model. The motion once generated according to the model can not be easily revised for a new sequence, showing a slightly changed behavior or the same behavior in a slightly changed environment.

3.1.2. Implicit Behavior Modeling Structure

An explicit structure is used to model the physical connections between the subparts in a single object's body. Realistic motion can easily be produced if such a structure is found. Examples using this structure are the tree-like structure of articulated human figure, the subdivision structure of fractal geometry, and the muscle groups used in facial expression animation. With an explicit structure, a single object's motion, even if it has a large number of degrees of freedom, can be easily controlled. Structuring an object's motion is one way of eliminating the blind guessing of the "correct" control values for a motion.

When an object is placed in an environment, the object's motion is not just affected by its own model, but also by the surrounding environment. Even though the "connection" from an environment to an object does not physically exist, it can still be modeled to show the environment's influence on an object's motion. The structure for modeling the environmental "connection" can be more dynamic than the structure of a single object. To reduce the guessing in scene motion, this structure should be explicitly modeled to effectively control the dynamic behavior of an object caused by the environment. Currently, this structure is either missing or implicitly used in scene motion applications.

Currently, a predefined sequence model is used for animating the motion in an environment. The sequence tells exactly what behavior an object should perform at every time step and every location along the predefined path. The animation is produced from one action to the next, and from one object to the next, where each motion is produced in its own control space. No "connections" between objects and between objects and the environment are generally modeled and used in the motion control. Essentially, the same control concept and mechanism used for a single object's motion is used for the case of a scene motion. This control approach can eventually match the motion to a predefined sequence, but it leaves the efforts to the sequence itself as no environmental structures are modeled for the similar use in a different situation.

To illustrate this point, let's consider dance motion in a room environment. Suppose the room environment is surrounded by walls and with some chairs randomly placed in the room. In this room environment, two dancers are practicing their favorite dances, together with the behavior of avoiding walls,

chairs, and the other dancer in the room. Periodically, a dancer may decide to switch his dance pattern to a new one, and show fear or dislike towards one of the chairs for some reason. To animate this dance behavior in the room, a sequence is usually first predefined. Along the defined sequential path, the behavior of avoiding the walls, chairs, and the other dancer is planned at precise times and locations. The behavior of changing a dance pattern is also precisely planned in the sequence. The fear or dislike behavior towards one particular chair is modeled in the sequence when this chair appears on the path. Now, assume that for some reason this behavior needs to be slightly changed, such as applying the fear or dislike behavior to another chair, or switching the dance pattern during another time interval. This slight change requires the redefinition of the sequence and the motion is reanimated from the beginning. There is no easy way to use the previously modeled behavior in a slightly different manner. The same is true for a slight change in the environment. For instance, if the chairs in the room are relocated, a new behavior sequence needs to be produced, recoded, recompiled, and retested to match the relocated objects. Notice that in this case the behavior is not changed, but the environment has, which also changes the precise time and location assumed in the previous sequential behavior.

3.1.3. Indirect Environment Control Effect

From the previous two subsections, we can see that additional control degrees for a scene motion are introduced by the environment. This additional control complexity increases with the number of objects in the environment and the behaviors modeled between these objects. The complexity also increases with slight variations in the environment, as no behavioral structure is explicitly modeled in the motion. With the missing environment structure, the animation of a predefined sequence relies on blind guessing to match the motion to the planned sequence. To search for a solution, we first ask whether these problems can be handled using the current techniques, which are aimed at providing a better motion control environment. For this purpose, we look at three approaches for reducing the control complexity of animation. These are motion representation and understanding [Esakov89, Fishwick88, Kalita89, Magnenat-Thalman83], three

dimensional interactive environments [Chen88, Takemura88], and command driven systems [Girard85, Zeltzer82].

Motion representation and understanding provides the user with a high-level control interface, which allows descriptions of knowledge, reasoning, level representations, and natural language. This approach suggests a great improvement over previously existing movement representations, such as labanotation, graphical languages, or robot control systems. One example of using a natural language interface is to characterize complex activities by English verbs that can be further specified using adverbial modifications of the motion verbs, such as a "more" or "less" modifier. Using the power of natural expression, this approach is expected to describe subtle movement, behavior, emotion, and intended movement.

Three dimensional interactive environments address the same problem, but suggest a different solution. This approach offers the user the ability to specify a motion directly using three dimensional interaction. The environment can be supported using conventional two dimensional input devices or three dimensional input devices. Research by Takemura [Takemura88] shows that a 3-D input device performs better than the 2-D input device in terms of task completion time and error rate, where the experiments used both a 2-D mouse and a 3-D magnetic tracking device. More research results on the use of three dimensional input and output devices appear in the work of [Fisher86, Green90], which shows simple object manipulations and navigation in a three dimensional environment.

Command driven animation systems provide the user with a ready-to-use system with a limited set of motion actions. These systems include facial expression, legged locomotion, or skeleton walking models. Most of these systems are based on animating a single object having a large number of degrees of freedom. Since the control details are handled by the system, the user does not need to worry about the control details. These systems release users from the burden of a complex animation; but the users have little say over how the motion will be performed, even though changing the parameters in the model is possible.

Overall, these three approaches reduce the control complexity by providing the user with a higher level interface. However, underneath these interfaces, the difficulty of the detailed control of a motion still

remains, including the additional control complexity introduced by a scene animation. The use of these interfaces shifts the problem of animation complexity to the system, but does not solve the problem.

The control complexity of a scene animation not only comes from the objects which contain a large control basis, but also from the environment where the motion is performed. The control degrees of freedom introduced by the subparts of a single object are handled by using the subparts structure. But, for the environment "connection" in a scene motion, no explicit structure has been modeled and used in the motion control process. As the environmental structure of a scene motion is missing or implicitly used in a predefined sequence, it is very difficult and expensive to model motions that behave independently in environments. This difficulty still exists even when the problem is shifted to the system underneath its interface.

3.2. Research Motivations

To effectively address the problem of a scene motion, we need to study the special control issues introduced by the environment and apply new concepts and mechanisms that can effectively deal with this problem. We have developed a new control model for the problem of scene animation that supports an intuitive, creative, and structural control environment for animation. The model should reduce the degrees of freedom introduced by an environment by directly addressing the problem, extend the animator's efficiency for animating scene motions by suggesting general solutions to animation problems, and explore the wide range of dynamic behaviors by providing explicit behavioral structures.

Several basic issues in scene animation are addressed by the new control model:

-- a new control concept for scene animation called relations,

-- new control mechanisms based on this new concept,

-- explicit control structures for using the new control mechanisms, and

-- a new behavior animation system using the new concept, mechanisms, and structures.

The first step in effectively addressing the additional control complexity of a scene motion is to use the new control concept. This concept is based on the general issue of how a moving object is connected to the environment, rather than how an object moves along a predefined path. If the general issue for controlling a scene motion is well understood, the solution to a specific motion can easily be derived as one specific application of this concept. On the other hand, if the solution to a specific motion sequence is produced, the solution can't easily be generalized to other specific motions.

The general control issue of a scene motion is first discussed from the environment side. The questions of how an environment restricts or stimulates a scene motion, and how a natural behavior is derived from the environment are asked, based on a general environment control perspective. This perspective in our study is based on an object-to-object control concept, which decomposes the control of a scene motion into the influence of each object in the environment. For example, consider a room environment with walls and three randomly located chairs. The control of the motion in this environment can be decomposed into the effect of each object in the room, such as each of the walls and the chairs.

From the object-to-object control concept, the influence of one object on a moving object can be separately modeled. In this way, the question of how to animate a sequential behavior in an environment is transferred to the question of how to organize the reactions between the objects in the environment. For the previous room example, the minimal environmental influences on a motion in the room are the avoidance behavior for each of the walls and chairs. Other alternative behaviors involving each of the walls and chairs can be similarly modeled from the object-to-object control concept. This concept defines the general view of scene animation, and the use of a set of controls modeled from this concept produces a specific motion application.

With the object-to-object control concept, two basic control issues need to be dealt with in a scene animation. One is the modeling of object-to-object influences, and the other is the combination of these influences to produce more complex behaviors. Modeling an object-to-object control is localized to two objects, the object presenting an environmental "connection" and the object responding to the "connection".

One response to one environmental "connection" between the two objects is modeled as one influence control. The description of a control should directly state the two objects that are connected, how the "connection" is sensed and the behavior between the objects. This description can be as simple as a high level language or a script language, based on the object-to-object control concept.

For a specific motion behavior in an environment, a set of controls should be selected and used to model the desired behavior. Behavior modeling is concerned with how to naturally integrate the controls during the progress of a motion, such as how to select a control and link from one control to the next. The modeling of a desired behavior can be produced using a high-level control interface, where decomposed controls from a scene environment are used as basic elements. One suggestion for such an interface is an interactive environment. Within this environment, the user can directly express how a desired sequential behavior should be modeled using a set of controls.

For an effective integration of controls, common structures used to link the controls are explicitly outlined in the integrated behavior. These structures are logically separated in several hierarchical levels. From the bottom level to the top level, simple behaviors are gradually integrated to a more complex and responsive behavior. The use of these structures reduces the cost of guessing a "correct" scene motion, whose environmental structures are unknown or implicitly used. Both behavior creation and revision can benefit from the use of structures, with simple structural editing such as an adding, deleting, or replacing of controls. This allows a wide range of behavior applications to be produced from a set of controls.

A behavior animation system using the new control concept, mechanisms, and structures proposed for scene animation is outlined. The system has the ability to describe the controls of a scene motion based on the object-to-object control concept, and integrate the set of controls to produce a desired environmental behavior, through an explicit structure approach. The system allows the user to interactively compose an environment and compose the sequential behavior in the environment at the same time. The use of the system motivates the general modeling of a scene motion, rather than a specific predefined path. The system sets an example which extends the traditional boundary of a complete programming or command interface

for a predefined motion.

3.3. Previous Approaches

In the extended problem domain of scene motion, the problem is usually handled according to a predefined sequence using the same control concept and mechanism as for a single object's motion. This predefined sequence model can not effectively handle the special control issues introduced by an environment, such as the large control basis, motion interactions, and dynamic changes in the environment and behavior. The implicit control of these effects makes scene animation a very expensive and time consuming task. The motion modeled by this approach is rigid, with little adaptive ability for slightly different environments and behaviors. These limitations are due to the lack of direct control and mechanisms which can support such a motion.

In addition to the predefined sequence model, a few other approaches have been proposed for the scene motion problem over the past decades. These are the sensor-effector approach, the rule-based approach, and the predefined environment approach. Detailed discussions of these approaches are included in the following subsections, which emphasize the inputs, outputs, and link structures between the two ends. The issue of how effectively each approach handles the problems of scene animation is also discussed.

3.3.1. Sensor-effector Approach

The sensor-effector approach is one of the first approaches to the problem of scene animation. This approach presents a control model which consists of three parts -- sensors, effectors, and a neural network which connects the sensors and effectors. Using this model, an object's motion depends on how the environment is sensed and how the sensed information is processed through the neural network. The output signals from the network are used to trigger various effectors, which produce the object's motion. The approach proposed by this model simulates the way that humans and animals usually perform in the real

world. The sensor-effector model is outlined in Figure 3.1.

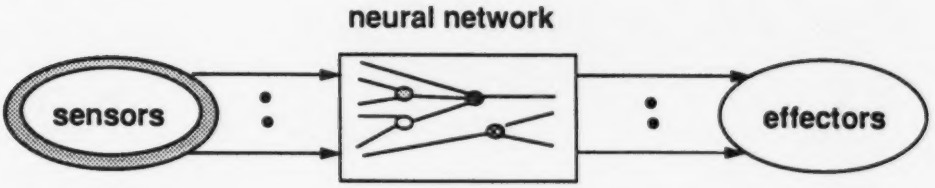


Figure 3.1 Sensor-effector Control Model

One important feature of the sensor-effector approach is its natural way simulating motion in an environment. It starts by sensing the environment, then some functions of the sensed information are used to trigger the object's effectors. However, the use of the model depends on the understanding of real neural networks. The efforts to explore such a potential have progressed over the years. But, it is still a research problem to determine how a neural network is actually connected between the sensors and effectors in a moving object in the real world.

One significant work in the sensor-effector approach is Braitenberg's book "Vehicles: Experiments in Synthetic Psychology" [Braitenberg84]. This book explains the possibility of generating some interesting motion behaviors, such as fear, aggression, love, selection, and foresight, in a simple environment of a light source and a few vehicles. A vehicle is modeled by sensors, motors, and some connections between them. Once a sensor detects the light source, the amount of sensed light is transmitted through the connections to motors which drive the object. Different behaviors can be produced by using direct or crossed connections, or other complicated layer functions. In addition, both excitory and inhibitory responses can be modeled in the connections. It is interesting to observe that different motion behaviors can be produced by such a simple connective model, which are simulated by wires and mechanical parts in the experiments.

BrainWorks [Travers88] is a system based on the ideas of Braitenberg, in which an interactive graph-

ics interface is used to construct a nervous system of a simple animal. Using this system, the user initially starts with a brainless animal body shown as a turtle, which is later equipped with visual sensors and touch bumpers and simple motors to drive the turtle in either position or orientation. The system allows the user to interactively select a nerve from a menu and a tool for connecting the nerve to the neural network. The network can be gradually built by repeating the interactions. Once the network is built, the turtle can display reasonable behavior, such as seeking or avoiding in an environment. One use of the system is to provide an educational environment for grade-school children to learn the basic behavioral mechanisms and autonomous animal motion through their own experience.

Inspired by the ideas of Braitenberg, Wilhelms [Wilhelms89] has proposed another interactive system for the control of behavioral motions. This system provides the user with an interactive control environment to map a connection network between sensors and effectors. Among the connections, various nodes can be used to invert, emphasize, or apply a threshold to the signal. More sophisticated procedural operations for other mappings can also be used. At the output end, the motion effectors are modeled by the forces whose magnitude and direction are determined from the output of the network. A positive valued effector produces a pushing force and a negative valued effector produces a pulling force. Using this system, the behavioral motion can automatically be produced by setting up the control network of transfer nodes and letting the motion go. As a result, examples of the attraction and avoidance behaviors among blocks have been produced.

3.3.2. Rule-based Approach

The rule-based approach is another solution to the problem of scene animation. In comparison with the sensor-effector approach, this approach uses similar input and output parts, taking the sensed information as its inputs and motor controls as its outputs. In between the inputs and outputs, this approach uses a set of behavioral rules to map from the sensors to the motors, instead of the neural network used in the sensor-effector approach. The control model of the rule-based approach, in terms of the three parts, is

shown in Figure 3.2.

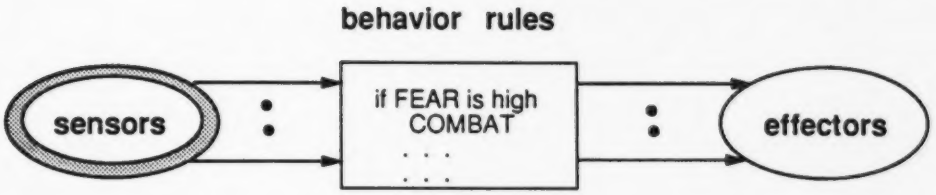


Figure 3.2 Rule-based Control Model

This approach uses a set of behavioral rules to determine the proper motion, such as when and what actions should be produced. The choice for selecting actions can be represented in a decision tree, where each branch contributes one control alternative. The decision amongst the alternative branches ranks the order of importance for selecting a control, depending on the weights and thresholds that are used by the behavior rules.

The use of behavior rules offers a great challenge for modeling a broad range of motions which are defined by a set of rules. But, the use of rules is less efficient due to the rule interpretation process, which travels through the rule decision tree. This inefficiency becomes worse when a large set of rules is used. Especially for the motion in a dynamic environment, it can be more difficult and less efficient to build and process a large decision tree, which covers all the possible behaviors in the dynamic environment. Updating a large decision tree due to a slight environment or behavior change can also be very difficult.

One noticeable work in the rule-based approach is Petworld [Coderre88], based on a world of pets, rocks, and trees which inhabit a two-dimensional environment. This research uses behavior rules to model some simple animal behaviors in the environment. Within the environment pets can move one unit along their body orientations or change their body orientations to a new heading. Pets have a field of view about 90 degrees, can carry one rock at a time, eat trees as food sources, use rocks to build nests, attack each

other, and die from starvation or wounds.

The pets' motion is basically performed in a loop of see-think-do. From the information perceived from the world, each pet has a chance to decide what to do before an action is performed. If more than two possible actions are suggested at the decision time, the selection is made based on the ranking of the actions. Conditionally, the one that has the most urgent priority is selected from the suggested actions. In the Petworld system, a decision tree covering all possible actions is built, where each branch of the tree represents one control alternative on the node conditions. For instance, at one node for the combating condition, both "attack" and "run" control alternatives are represented as two branches.

The selection at each decision node is expressed by a few behavioral rules. One example of a behavioral rule used at the combating node is:

Combating

1. If you have an available attack, and your damage is low, then recommend a tradeoff of attacking and running away.
2. Otherwise, recommend running away from any visible pets.

The ranking of an action is computed according to each pet's internal states such as HUNGER, FEAR, INJURY, and others, which can be interactively varied through the system's interface. In addition, several decision strategies can be applied to construct the competitive, compromise, and displacement behaviors.

A model of flock-like behavior based on a centralized, noncolliding aggregate motion has been produced by Reynolds [Reynolds87]. In his study, the behavior of each bird in a group is simulated independently according to the bird's local perception of the dynamic environment. The model is first supplied with a geometric form and the ability to fly; then the behaviors of a flock, such as avoiding a collision and the urge to join the flock, are applied using the proper forces. The behaviors are abstracted to rules in the precedence order of collision avoidance, velocity matching, and flock centering. These behaviors model the ability of individual birds to fly and participate in natural aimless flocking. The flock's motion can be

directed by controlling a global position vector or a global direction vector for the flock. In general, the grouping principles behind this behavior model can be used for other kinds of aggregate motion, such as a herd of land animals or a school of fish.

3.3.3. Predefined Environment Approach

Another approach to solving the problem of scene animation is based on the assumption of a predefined environment. As the environment is previously known, a specific motion behavior in the environment can be carefully planned. One typical application using this approach is to select one optimal path in the environment, either the shortest path or the path using minimal energy for the moving object. This minimal path is derived from all the alternatives starting from an initial position to a goal position in the environment. The control model proposed by this approach is outlined in Figure 3.3.

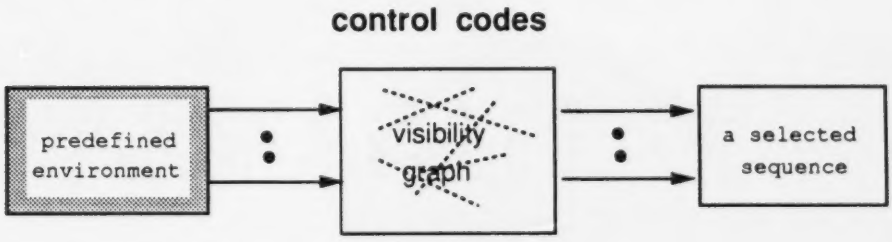


Figure 3.3 Predefined Environment Control Model

This approach has the potential to control a complex motion or motions in an environment, when the environment is previously known. Based on the environment, a scene motion which satisfies certain behavior criteria can be carefully controlled. It is also possible that the controls of a scene motion can be automatically produced by the animation system, when the user picks the initial and goal positions of a motion from the environment and selects the behavior criteria from the system's menu. However, the motion produced by this approach depends on the predefined environment. There can be a high cost associ-

ated with slightly varying the environment by adding, deleting, or relocating objects in the environment.

Path planning in a static environment is the typical application of the predefined environment approach. This application starts from a given static environment and then the motion (or motions) which can navigate around the obstacles in the environment are planned. A well-known technique used for this type of applications is the visibility graph for the obstacles in the environment. According to this graph, a collision-free minimum-cost path can be found from an initial position to a goal position, among the paths depicted in the graph. The use of a path planning algorithm has been extended from the robotics applications to behavioral animation. The path planning approach focuses on converting the piecewise linear paths, that current algorithms generate, to smooth paths and accommodating trajectory planning to the regions predicted in both the space and time domains.

The work of Ridsdale [Ridsdale88] suggests the use of knowledge-based systems for planning the motion in a stage environment. He proposes a system which has the ability to select an appropriate path from position A to position B with respect to the other characters present on the stage. If another character appears on the path initially selected between the two positions, rules are used to search for a "good" revised path which can avoid the obstacle and also keep the same attitude to the other characters on the stage. This system uses a stage database which may be updated each time a character on the stage moves. The updating process increases the control overhead of the animation. Also, examples shown by the system are mainly the motions that are planned while only one object moves on the stage. The question of real-time planning for a motion in a dynamic environment, where several objects move at the same time, still remains unsolved.

3.4. Proposed Relation Approach

To effectively address the problem of scene animation, a new control model is proposed. As in previous approaches, this model takes sensors as its input and effectors as its output, where the sensors are based on the four common sensing channels -- visual, smell, sound, and tactile. Behaviors are generated by procedural control primitives that produce natural movements for the object. These control primitives are called relations and are individually triggered by the sensed information and combined to produce more complex behaviors. This model is called the relation model and the approach using the model is called the relation approach. This model is illustrated in Figure 3.4.

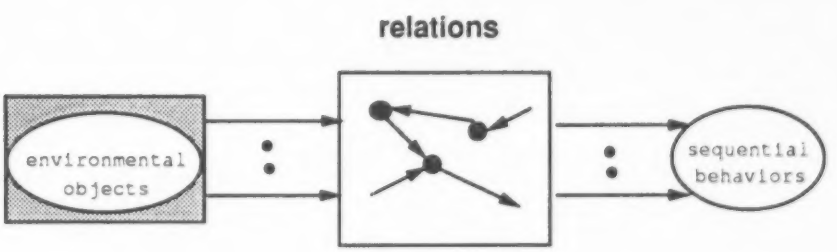


Figure 3.4 Relation Control Model

The relation model is based on the new control concept of scene animation. Since the motion is influenced by the environment in many ways, we propose that motion control should be decomposed into primitives which individually address the relevant environmental influences. These influences should be modeled directly and individually, which supplies the necessary basis for further systematic control of these influences. The decomposition of motion control is based on the object-to-object influence control concept, which we call relation control. Each relation models one interaction between two objects, one presenting the source and the other responding to the source. A relation shows one response which can dynamically participate in a motion, depending on the behavior modeled by the relation. Using alternative relations, an animation can easily be modified to produce another behavior or used in another slightly different environ-

ment. A formal definition of a relation in the context of scene animation is given in Chapter 4.

We expect that the control environment supported by the relation model will provide an intuitive, creative, and structured working space for scene animation, in a wide range of behavior applications. Within this environment, the users have the ability to freely express their ideas on the behaviors in a scene motion, rather than following the examples provided by a system. Several important steps in developing the components of the relation model are:

[1] relation definition and its theory

The first step in defining the relation control model is to precisely define the concepts of an environment (static or dynamic), an environmental influence, and a relation. The diversity of relations used in the problem domain of scene animation is divided into a few types. Based on these types, common control properties of relations are defined and discussed. Other control issues of relations, such as the environmental mapping and minimal and maximal uses of relations in general, are studied.

[2] frame modeling of a relation

For the modeling of relations, a general relation frame is used. This frame includes both the local control properties and the main reaction process of a relation, where the local properties are separated into suitable property description and interactive control access. The main reaction process describes the detailed control of the behavior. The use of relation modeling frame is independent of a specific language or a specific control technique.

[3] structural synthesis of relations

Individual relations are structured while modeling a desired dynamic environmental behavior. The structures that can be used among relations are organized in a relation synthesis hierarchy, with multiple control levels. Each of the hierarchical levels addresses one important behavior control aspect. The use of the hierarchy allows easy addition, deletion, and replacement of relations or other structuring mechanisms which can clearly outline the interconnection of the relations in a modeled behavior.

[4] behavior editing

The structures built in a scene motion are not just used for one behavior application, but for exploring a wide range of behaviors in the environment by revising the built-in structures. The structural interface through the levels of relation synthesis hierarchy directly supports the behavior control aspects, which can be easily located for the behavior revision purposes. A behavior revision can be directed either to the highest control level of natural behavior expression or to one of the lower levels localized for a partially connected behavior structure.

[5] prototype implementation of the relation model

Since the relation control approach introduces new concepts, mechanisms, and structures for animating a scene motion, a prototype implementation using the ideas of the relation control approach has been produced. This prototype implementation example is expected to provide an intuitive, creative, and structured control environment, which completely differs from the traditional sequence approach directly copied from the problem domain of a single object, as well as other proposed approaches. Both scene environments and dynamic environmental behaviors can be composed on-line using the implemented system.

It is desirable to organize the system into various interface levels which can serve users with different knowledge and experience using the relation control model. The naive user can issue a named motion behavior recorded by the system. The user with limited relation knowledge can experiment with one particular example modeled by the system to see how the behavior is structured and learn to revise the behavior. The experienced user can directly use the system facilities to model a new behavior, by describing the new relations and structuring the relations from the lowest control level to the higher levels.

3.5. An Example Using the Relation Approach

Consider a walking motion in a room environment as an example using the relation approach. In this example, the environment consists of walls, three chairs, and two persons, which are randomly placed in the room. The behavior that we are interested in modeling is as follows: Each of the persons initially selects a heading for a walk. During a walk, a person tries to avoid the chairs randomly placed in the room, the other walking person, and the room boundaries. If a potential collision with one of the chairs, other person, and room boundaries is detected, the person will take a necessary response to avoid the potential collision. One of the persons may also express a dislike or fear behavior while avoiding one of the chairs, which is covered with a particular color or texture pattern or just identified by the person.

To model the above behavior by using relations, the motion is first decomposed into individual relations based on the object-to-object control concept. There are three basic object-to-object mappings in the room environment, which are: wall-to-person, chair-to-person, and person-to-person. Here, a group of objects with similar control properties, such as walls, chairs, and persons, is viewed as one object entity in our relation decomposition scheme. Based on the three mappings, six relations are used to model a person's walking behavior in the room:

relation	object-to-object:	behavior control

walkstep	person_i-to-person_i:	move one walking step
avoidperson	person_j-to-person_i:	turn away from person_j
avoidwalls	walls-to-persons:	turn away from a too-close wall
avoidchairs	chairs-to-persons:	turn away from a too-close chair
dislikechair	chair_2-to-person_2:	reverse turn at chair_2
fearchair	chair_3-to-person_1:	take few steps back at chair_3

The "walkstep" relation is mapped from and to the same person, where the variable person_i denotes one of the persons. "Avoidperson" is mapped from one person to the other, where the variables person_i and person_j denote two distinct persons. "Avoidwalls" is mapped from the walls to the persons, where both objects are groups. The control of this mapping is applied for each of the persons against each of the walls in a double control loop. Similarly, "avoidchairs" is mapped from the chairs group to the persons group. "Dislikechair" is mapped from chair_2 to person_2, where the index value 2 denotes the 2nd chair and the 2nd person in the group. "Fearchair" is mapped from chair_3 to person_1, using a similar member-indexing function.

Each relation models one simple behavior triggered by one environmental influence mapped between the two related objects. Brief descriptions of the local influence and behavior of the above six relations are: "walkstep" advances the current walking step to the next step. "Avoidperson" produces a turning response when the person views the other person coming too close to the path. Here, the relation is triggered by a distance threshold mapped between the two walking persons. "Avoidwalls" turns the person when he/she is coming too close (a distance influence) to one of the walls, to a new heading in which the next two walking steps can be produced. "Avoidchairs" turns the person away from a chair that is too close to the person. "Dislikechair" produces a quick reverse turn when person_2 is close to chair_2, where both the distance threshold and chair_2 identification are the unique sources for the relation. "Fearchair" results in person_1 taking a few backward steps, when the person gets too close to chair_3.

Each relation performs a simple behavior triggered by an environmental influence. Whenever the influence becomes active, the behavior is automatically triggered to its active response. For instance, the relation "walkstep" is triggered active when the person is motivated to walk. This active relation produces a

simple walking behavior step by step at each motion control step. Relation "avoidwalls" is triggered active if the distance from the person to one of the walls is below a threshold, which results in an active avoiding behavior to lead the person heading away from the wall. When a person walks in the room, his/her motion varies from one behavior to another, or to some complex behavior composed of several simple behaviors. The change of behaviors in a scene motion depends on the dynamic influences of the environment and the control structures imposed on the use of relations. Here, we assume that the relations are separately modeled for each described behavior. Among these relations, certain control structures are also modeled in a way described below for the desired walking behavior in the room.

Initially, each of the persons is self motivated to walk in the room. This motivation triggers the relation "walkstep" to its active response, which produces a walking step at each control step. This simple walking behavior can be interrupted by the other relations, if a wall or a chair in the room appears too close to the person. The interrupting structures among relations are modeled by the relation state controls, whose details are explained in Chapter 5. At this point, we assume that these interruptions are possible and consider only the effects produced by them. For instance, when the relation "avoidchairs" is triggered active by its distance threshold for one of the chairs, the relation interrupts the active walking relation and blocks it while avoiding the chair. This relation stays blocked until the avoiding relation becomes inactive again. Similar interrupting structures are modeled to block the active walking relation when a distance threshold is sensed from either one of the walls or the other person walking in the room.

The use of two alternative relations "dislikechair" and "fearchair" requires additional interrupting structures. Consider the case when the distance threshold is detected from the 2nd chair in the room, and both relations "avoidchairs" and "dislikechair" are simultaneously triggered active. At this time, the relation "dislikechair" interrupts not only the walking relation, but also the avoiding relation commonly used for all the chairs in the room. It blocks the walking relation during the dislike behavior period and cancels the active state of the common avoiding relation. After the blocking period, the walking relation is resumed to its active behavior. A similar analysis can be produced for the "fearchair" relation. But, this time a different

interrupting structure is used from the "fearchair" relation to the "avoidchairs" relation. When these two relations are triggered active by a distance threshold from the 3rd chair to the 1st person, the relation "fearchair" blocks both of the relations during its fearing behavior. After that, both relations are recovered to their active states. From there, the "avoidchairs" relation again blocks the walking relation during the period of its normal chair-avoiding behavior. The possible interrupting structures among the relations during the walking behavior in the room are depicted in Figure 3.5, where each structure can be dynamically introduced at one instant in time during the motion. Here, the solid arrows predict the possible interactions among the necessary relations used for the walking behavior, and dashed arrows predict the interactions among the alternative and necessary relations.

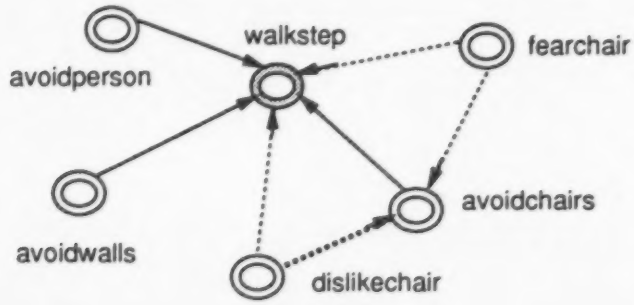


Figure 3.5 Relations Interrupting Scheme While Walking in a Room

Note the potential changeability of a modeled behavior by using the relation control approach. This ability can be illustrated from the above walking example. To vary the walking behavior described above, we can simply map the "dislikechair" relation to another chair member, or extend the "fearchair" relation to all the chairs in the room after a few times of chair-avoiding experience, or add, delete, and replace a relation, or structure a different type of interruption between the two relations. There are many ways to simply change the behavior modeled in the walking motion, based on the decomposed relation controls.

Chapter 4

Theoretic Foundation of the Relation Model

This chapter gives the formal definitions of environments (both static and dynamic), a response, an enabling condition, a relation, a directed graph of linked relations, and a sequential behavior. Four control properties of relations are discussed, which are: optional, interactive, selective, and variable structuring properties. The mapping of related objects in an environment, including both an individual and a group of objects which may be environmentally mapped to one or more relations, is depicted. The diversity of relations used for a scene animation is divided into a few class types, based on the differences between the object sources. The use of relations in terms of the two extreme cases -- the minimum and maximum numbers, is also analyzed.

4.1. Definitions of the Relation Model

There are two types of environments found in the problem domain of scene animation: static and dynamic. A static environment is composed of several static objects, one moving object, and an area restricted by the environment boundary. A static environment may also involve several moving objects which do not interact in either the time or space dimensions; each object can move either in its own space, or they can move sequentially in the same space so their movements do not overlap in time. This special case, with multiple moving objects whose motions are independent of one another, is viewed as the equivalent of one moving object.

A dynamic environment is composed of several static objects, several moving objects, scene events that dynamically occur in the environment (such as a sound), and an area restricted by the environment boundary. The motions of objects are not isolated, but interact with one another at the same time and in the same area. These motions can also interact with events occurring in the environment at uncertain times.

The use of static environments in scene animation forms a large and currently popular animation application domain. Static environments are similar to dynamic environments, but are easier to describe.

Since there are a large number of important applications that can be handled by static environments, they form an important subset of dynamic environments. Dynamic environments are applied to the more general research issue of scene animation.

The definitions of the two environment types used in scene animation are:

Definition 1

A static environment (SE) is an environment composed of one moving object or several moving objects whose motions are independent of one another, several static objects, and a bounded space; whereas, **a dynamic environment** (DE) is an environment composed of several moving objects whose motions are dependent on one another, several static objects, scene events, and a bounded space. These two environment definitions can be symbolically described as:

$$SE = (M, \{S_j\}, B) \qquad \textit{where } j=1,\ldots,m,\ m\geq 0$$

$$DE = (\{M_i\}, \{S_j\}, \{E_k\}, B) \qquad \textit{where } i=1,\ldots,n,\ n\geq 2;\ j=1,\ldots,m,\ m\geq 0;\ k=1,\ldots,q,\ q\geq 0$$

where M or M_i is a moving object, S_j is a static object, E_k is an event, and B is a bounded space.

An example of a static environment is a room (the environment boundary) with a number of chairs (the static objects) and one walking person (the moving object). An example of a dynamic environment is a room with a number of chairs, several people walking around the room, and irregular occurrences of sound. In this example, the walking persons are the moving objects and the sound occurrences are the events occurring in the room environment. The static objects and the environment boundary in this particular example are the same as for the previous static example.

Since the problem domain of scene animation includes both static and dynamic environments, the terms "static environment" and "dynamic environment" will be heavily used in the following discussions. For simplicity, the word "environment" will be used to refer to static environments and the words "dynamic environment" for dynamic environments.

An object's motion in an environment can be decomposed into a set of control units based on environmental influences. These influences can be traced to the environment boundary, obstacles, other static and moving objects, and scene events. Each of these influences can be dynamically enabled, disabled, and systematically developed during a motion. While a motion progresses in the dynamic environment, the motion can be dominated by one major influence at one time, but collectively controlled by several influences at another time. The active set of environmental influences can be dynamically converted to different sets during a scene motion. For instance, when an object moves closer to a table, the object should naturally respond to the close distance to avoid possible collision with the table. At this time, the motion is dominated by the table influence. After avoiding the table, the object may turn its head to another passing object and shortly stop or slow down its motion because of the occurrence of a sound. At this time, the motion is collectively controlled by both another passing object and a sound event.

From time to time, a scene motion can be dynamically formed by a varying set of environmental influences, including newly introduced ones, continuously performing ones, and other temporarily blocked or resumed ones. If these dynamic influences can be properly controlled in a scene motion, a rich set of desirable and natural motion behaviors can be modeled. One example using a variable set of influences is the motion of rolling balls through a room environment. In this dynamic situation, various avoiding influences by the balls can be produced depending on what strategies and control mechanisms are used. Each environmental influence produces a response. The formal concept of response with respect to the domain of scene animation is defined as:

Definition 2

A response r is a primitive unit of a motion determined by one environmental influence and can be described by a tuple (c,t,st,w) where c is the control process that executes the response, t is the response duration, st is the response strength, and w is the set of switches for determining subtle differences in response.

One simple example of a response is a look response. The look response is modeled by the control process that generates the primitive motion of looking, the duration of the look, the strength of the look (such as the head and body turning speed and final orientation), and optional control based on other environmental conditions (see section 4.2 for examples). If we consider as our environment example a room with chairs, two walking persons, and irregular sound events, the possible responses in this environment can be a walking step, a side step, a forward step, a backward step, a jump, a pause, a body turn, an arm raise (left or right), a nod, a clap, and a grab.

A response can be repeatedly used in a motion any number of times. During the motion, a response could behave as a candidate ready to be triggered, a participant actively reacting in the motion control process, an idle actor temporarily blocked by other responses, or an unselected candidate whose effective control status has been canceled from the current control session. These different reaction stages are characterized by the four control states: *potential, active, suspended, and terminated.* At any time, a response must be in one of the four control states. A formal description of state control is included later in the relation definition in this section.

The control process of a response can be locally varied by its duration, strength, and switched optional control. A response's duration determines the length of time that the response should last; a response's strength determines the amount of the response; and a response's optional control determines the possible response adjustment based on other environmental conditions.

The active state of a response is triggered by an enabling condition sensed from the environment, which can be based on an object's color, size, or velocity, or the distance to an object. The enabling condition of a response can be in general based on one or more sensory properties, such as the geometrical, physical, optical, or other properties, sensed through the visual, sound, smell, or tactile channel. The use of an enabling condition explicitly describes the environmental "connection" between a response and an environmental influence sensed in the environment. Here, the enabling condition is the source of the response, and the response is the effect. The formal definition of the enabling condition of a response is:

Definition 3

The enabling condition C_o of a response is a Boolean expression involving one or more sensory properties of an object o, sensed from either the visual, sound, smell, or tactile channel.

For the previous room environment example, the possible enabling conditions in the environment are Boolean expressions involving the size of a chair, the color of a chair, the distance to a chair, seeing a chair, seeing a wall, seeing the other person, a specially textured surface of a wall, the walking speed of the other person, the distance to the other person, and hearing the sound. These conditions can be sensed as the sources upon which a number of responses can be triggered.

A relation provides a mapping from the object presenting an enabling condition to the object performing the conditionally triggered response. Between the two related objects, a relation also describes the details of how the condition is sensed and what response results from the sensing. For example, a "friendly" relation can lead to a smile (response) from a moving object to the object identified as a friend; a "strange" relation can result in a slow motion response from the object appearing as a stranger. Besides the direct control related from a perception to a reaction, a relation's state is another control used to determine whether a relation should be used or not in each dynamic environment situation.

A relation has three main parts: the source, the responder, and the response. A relation's source is the object presenting the enabling condition, the responder is the object that performs the reaction triggered by the enabling condition, and the response describes how the responding object changes. The source object of a relation may be either a static or a moving object, while the responder object may only be a moving or a dynamic object. A diagram illustrating the two related objects and the response mapped between the two objects is shown in Figure 4.1.

The enabling condition plays an important role in connecting the parts of a relation, and thus is explicitly stated in a formal definition of a relation. Also, as relation state control addresses the important issue of other indirect environmental influences on a relation, it is also explicitly expressed in the definition. The formal definition of a relation is:

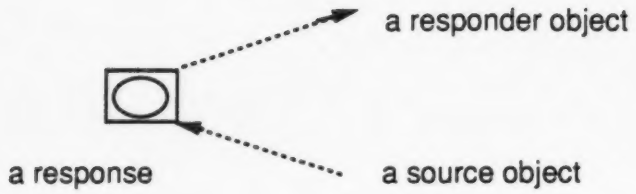


Figure 4.1 A Response Between the Two Related Objects

Definition 4

A relation R is a tuple (O_s, O_r, C_o, r, s) where the O_s denotes the source object or a group of similar source objects, O_r denotes the responder object or a group of similar responder objects (the source and the responder could be the same object), C_o is the enabling condition sensed from the source, r is the response that the responder performs, and $s \in (S_{potential}, S_{active}, S_{suspended}, S_{terminated})$ is the control state of the relation. Each of these states respectively signals either the ready, active, blocked, or terminated state of a relation.

One example of a relation in the room environment is the "avoid_chair" relation between a chair and a walking person. In this relation, the chair is the source object, the walking person is the responder object, a close distance to the chair is the enabling condition, and an avoiding primitive motion such as a turning is the response. Another example is the "like_person" relation between two walking persons, where the liked person is the source object, the other person is the responder object. In this example, the enabling condition is that the liked person is in the other person's sight, and the response is a raised arm showing the "like" feeling.

Among the four relation states, $S_{potential}$, S_{active}, $S_{suspended}$, and $S_{terminated}$, the active state is the only one under which a relation can perform its response. Once a relation is switched to the active state, its response process is actively controlled by the other control properties, such as response duration, response strength, and the use of optional switches. These local control properties can be adjusted every time a

relation is changed to the active state. Thus, an active relation can be identified not only by its active state but also by its local properties at the active time. Symbolically, an active relation can be described as:

$$R(O_s, O_r, C_o, r_i, S_{active})$$
$$r_i(c, t_i, st_i, w_i)$$

where r_i includes the local control properties, t_i, st_i, and w_i, at time i under the active state S_{active} of the relation.

More than one relation can be active at any time; these form the set of active relations and can vary in number. One such example is the walking motion in a room. While walking in the room, the person may wave to a friend passing by and approach a target at the same time. At another time the person may avoid a rolling ball appearing in its path or follow the ball's motion instead. This dynamic change of a scene motion can be modeled by the use of relations, depending on both the conditional sensing and the state control of the relations. The conditional sensing of a relation can cause a state transition of the relation itself, but the state control of a relation in general involves more complex environmental control factors, such as the use of control strategies, modeling of individual characters, and selections based on both urgent and personal reasons.

One of the reasons for a state transition is the interaction among relations. It is possible that a relation can be activated by another active relation, or an active relation can be temporarily blocked or terminated by another active relation. During the interaction among relations, we can view these state transitions as chains dynamically propagated from one relation to the next. A chain occurs when one relation causes a state transition in another relation, which causes further state transition in other relations. The relations whose states are changed in a chain-like effect form a directed graph of linked relations. The formal definition of the graph is:

Definition 5

Let $G = (V, E)$ be a directed graph of linked relations, where the vertex set $V = \{R_i\}$ is the set of relations, and the directed edge set E contains the set of $\{<R_i, R_j>\}$ where R_i controls the state

transition of R_j, such that

$$< R_i,R_j > = R_i(O_{si},O_{ri},C_{oi}, r_i, S_{active}) \rightarrow R_j(O_{sj},O_{rj},C_{oj}, r_j, s^*)$$

where R_i and R_j refer to two distinct relations, $<R_i,R_j>$ refers to a state transition from relation R_i to relation R_j, which occurs at some time during the motion, and s^* denotes the new state caused by the interaction.

A directed graph of linked relations can be divided into a set of sequentially linked chains, each of which forms a subgraph of a directed graph. A sequentially linked chain is restricted to consecutive connections between two adjacent pairs, such as $RL = \{ \cdots , <R_i,R_j>,<R_j,R_k>, \cdots \}$. With one exception, a relation cannot appear in more than one edge in a sequential chain. However, a cyclic chain can be formed as one special type of sequentially linked chain, with the first and last relations being the same: $RL = \{<R_i,R_j>,....,<R_n,R_i>\}$. Each directed edge in a sequential chain or a graph represents a state control issued at one instant during a motion.

Besides the graph of linked relations, individual relations can also participate independently in a motion. These relations are referred to as individual active relations. The reason for the state transition of an individual relation is the control of its conditional sensing and response duration. A relation can participate independently in a motion when its enabling condition becomes true; the active response of an individual relation ends when its effective response duration is over or when its enabling condition becomes false. Another reason for the state transition of an individual relation is due to relation scheduling control, which is introduced in Chapter 5.

During the period of a motion sequence (from the initial time T_0 to the final time T_n), groups of linked relations and individual active relations can participate dynamically in the motion at each point in time. If the *ith* time instant is denoted by T_i, where T_i satisfies the condition $T_0 \leq T_i \leq T_n$, then we count the groups of linked relations and individual active relations at each time instant T_i and collect them from the initial time T_0 to the final time T_n. These collected relations in a motion sequence describe the control complexity of the motion.

The sequential behavior of a motion can be stated as a collection of relations from the initial time T_0 to the final time T_n. The collection of relations includes both the groups of linked relations and individual active relations, which participate in the motion at one time T_i during the sequence period $[T_0, T_n]$. An individual relation is collected at time T_i if the relation is triggered active at that time. A linked relation, whose state is interacted by another relation, is collected at time T_i if it is active at that time. Note that a linked group (as a directed graph) may grow from one time to another. From these observations, we define a sequential behavior as a collection of relations at each time T_i during the motion, as follows:

Definition 6

Let $A(T_i)$ be the set of active relations at time T_i (including individual and linked active relations), where $T_0 \leq T_i \leq T_n$. A sequential behavior in the period $[T_0, T_n]$ is the collection of sets $A(T_0), A(T_1), \ldots, A(T_n)$ at each sequential time instant.

4.2. Special Control Properties of Relations

As stated in the previous section, a motion can be controlled by a set of relations. The control of the motion based on these relations is not merely a linear summation of the relations, but a more complicated dynamic process. The complexity of controlling relations to produce natural environmental behaviors can be attributed to the changes in the environment, interactions between objects and scene events, various motion habits, and personal preferences introduced by the object types in the environment. Another reason for the dynamic control of relations is the production of easily modifiable sequential behavior. Relations should be dynamically organized in such a way as to resolve potential conflicts, cooperate for similar goals, define individual behavior elements, and combine the elements to set desired behavior sequences.

One simple example showing the dynamic control process of relations is the motion of walking towards an attractive target in an environment with a number of rolling balls. In this environment, the motion towards the target may be delayed by avoiding the rolling balls, watching the balls, and following one of the balls with a specially textured surface. The motion may also be temporarily paused to yield to

fast passing balls, or completely replaced by a following reaction towards one attractive rolling ball.

This motion of the scene above can be decomposed into a set of relations which are mapped individually from the person to the environment boundary, to the rolling balls as either a group or an individual, and to the attractive target. Modeled individually, these relations can be dynamically linked in such a way as to convey the desired behaviors. For instance, if each response towards the rolling balls, as described above, is modeled by a relation, four relations named as "ball_delay", "ball_watch", "ball_follow", "ball_pause" can be used to model these behaviors. These relations are dynamically applied to the motion, independent of the ordering of the rolling balls and when the motion takes place. The relations involving the balls can also compete dynamically with one another and cooperate with the relation "towards the attractive target" during the motion.

The following is a discussion of relation control properties, especially the collective use of relations in dynamic environments. Four control properties of relations are outlined, which are: *optional*, *interactive*, *selective*, and *variable structuring*.

The *optional property* shows the subtle behavior differences of a response based on other environmental conditions. Such a condition is not the enabling condition that is directly specified in the relation. Instead, it is an additional one sensed from the current local environment, such as a nearby obstacle to the left. The influence of the other condition can be optionally controlled using switches. A switch is a variable which has two control values: "on" and "off". The "on" switch value adds the control to the response; the "off" switch value turns off the additional control. The change in response could be a slightly increased velocity or a directed turning reaction away from the nearby obstacle. For instance, the response of turning away from a wall could be a left turn if another obstacle is located on the right. The response of avoiding an obstacle could be accelerated if another moving object is approaching quickly.

The *interactive property* describes possible state transitions of a relation that are caused by the other relations. For example, a potential relation can become active, or an active relation can be temporarily blocked or terminated from the current motion control session. Here, the state transition of a relation is

caused by the interaction of another active relation. Different state controls can be issued from an active relation to other relations used in the current application. Each of them causes one type of state transition in the interacted relation.

If any pair of states in the state set ($S_{potential}$, S_{active}, $S_{suspended}$, $S_{terminated}$) is considered as a possible state transition, we have nine such cases out of twelve as the three paired from the state $S_{terminated}$ to others are not possible. These nine possible state transitions are listed in the following.

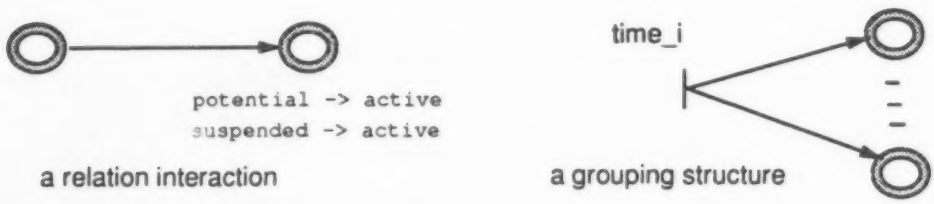
potential	-->	active
potential	-->	suspended
potential	-->	terminated
active	-->	potential
active	-->	suspended
active	-->	terminated
suspended	-->	potential
suspended	-->	active
suspended	-->	terminated

These nine possible state transitions are analyzed using the two interaction forms: *parallel* and *sequential*. *Parallel interactions* are those in which both interacting and interacted relations are active at the same time, where the interacting relation issues a state control which changes the state of the interacted relation. On the other hand, in *sequential interactions* only one relation (either interacting or interacted) is active after a state-control interaction.

A parallel interaction by definition causes the state of the interacted relation to change from any state to only the active state. The possible states for such a transition are potential and suspended. The transition is caused by either activating a potential relation or releasing a blocked relation. The reason for issuing a parallel interaction is to add other relations which can assist the motion of the interacting relation. One example is the parallel use of two relations "approach an attractive target" and "look around", where the active looking reaction is added to the approaching relation which issues the parallel-form interaction.

Another control mechanism used for parallel interaction is a grouping, in which several relations become active at one instant in time. The grouped relations are active at the same time, when the referenced time is reached and their enabling conditions are true. This control mechanism is viewed as one special case

of parallel-form interaction, where the role of interacting relation is replaced by the referenced time. Detailed discussion of the grouping control mechanism is presented in Chapter 5. In the following diagram, three parallel interactions, in terms of the two possible interactive state transitions (potential --> active, suspended --> active) and one grouped state transition, are illustrated.



A sequential interaction by definition results in one of the relations, either interacting or interacted, being active after a state-control interaction. Here we only consider the case when the interacting relation remains active after a sequential interaction. This means the state of interacted relation is converted to a state other than the active one. There are seven such possible cases out of the nine if we ignore the two in the previous situation. When searching for the circumstances in which to issue these seven transitions, we focus on the final states produced by the transitions. There are three such final states (potential, suspended, and terminated), which divide the seven transitions into three groups. The transition group for the final potential state is: active --> potential, suspended --> potential; the group for the final suspended state is: potential --> suspended, active --> suspended; the group for the final terminated state is: potential --> terminated, active --> terminated, suspended --> terminated.

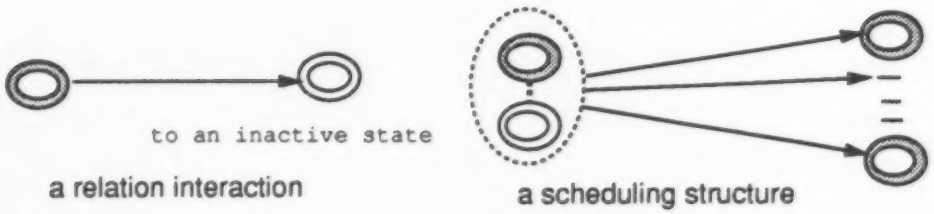
The transitions leading to the final potential state are issued when the response of the interacted relation is disabled. Such a transition indicates that the interacted relation is not currently active as is the interacting relation which was selected for more important reasons. When converted to the potential state, the interacted relation is still a ready candidate for the next chance. The transitions leading to the final suspended state are issued when the response of the interacted relation is blocked during the active period of the interacting relation. Such a transition is issued when the two relations, interacting and interacted, are

mutually exclusive or alternative relations. Only one of the relations (the interacting one in this case) should be in the active state. Transitions leading to the terminated state are issued when the response of the interacted relation is no longer used in the current motion session. These transitions cancel the relations mapped to the dynamic object sources which have left the environment. These sources can be another moving object or a scene event. When such a case is certain, removing these unnecessary relations can speed up the process.

One example of a sequential interaction is the use of two relations "avoid an obstacle" and "approach an attractive target". Assume at some point in the motion, the relation "avoid an obstacle" becomes active and it issues a state control to the relation "approach an attractive target". If the issued control is one of the transitions leading to the potential state, the relation "approach an attractive target" is disabled, but can be used in the next time step. The possible state transition of the relation depends on its state when the control is issued. For instance, the state transition "active --> potential" is produced if the approaching relation is previously in its active state. A similar case results in the other possible transitions. Alternatively, if the issued control is one of the transitions leading to the final suspended state, the approaching relation is blocked during the active duration of the avoiding relation. This blocked relation is recovered to its previous state when the blocked period is over. Also, if the issued control is one of the transitions leading to the final terminated state, the approaching relation is canceled from the current process as the target has already moved away from the scene environment.

In addition to the interactions between two relations, one or several relations can become active upon the active state transition of another relation. This scheduling structure can be viewed on the whole as another form of sequential interaction. Three time references of a relation's active duration can be specified: the first time step, the last time step, an intermediate time step. If relations are scheduled at the first time step they become active when the referenced relation becomes active. A similar rule is used for the final time step. For an intermediate time step, the scheduled relations become active at one time during the active duration of the referenced relation. In the following diagram, these sequential interactions, in terms of the

ones in the sequential interactions and three scheduling references, are illustrated.



The *selective property* of relations indicates the selection of one relation from a set of mutually exclusive relations. There are two reasons for such a selection: the order of importance and the order of personal preference. A selection of either reason can be determined by the assigned priorities, which could be a separated value range or a specially assigned priority set. As the priorities are assigned by reasons, they can be easily used to address one of the reasons. For instance, the personally assigned priorities can be used to model a changed mood or an accumulated time experience in response to the same environmental situation.

One example using the personally assigned priorities occurs when a moving object sees two attractive targets and hears a sound event at the same time. In this example, at least three relations "approach target 1", "approach target 2", and "pause at sound" become active by the enabling conditions sensed in the environment. However, as the three relations are contradictory to each other, the selection of one of the relations is necessary, based on the use of personally assigned priorities. For instance, if the object is timid, the object will likely stop at the sound and ignore the two attractive targets. If the object is determined, the object will continue approaching one of the targets and ignore the occurrence of sound. The selection of one of the targets depends on the object's personality.

Based on the priorities assigned either urgently or personally among a set of relations, a selection can be conducted using one of two strategies: the one-step strategy and the two-step strategy. The one-step strategy is applied directly to the priorities assigned to the relations, where the highest priority is selected. Alternatively, the two-step strategy is applied in two steps: one is the selection of sensing channels and the other is the selection of assigned priorities in each channel. One possible order of sensing channels is deter-

mined by the frequency of use of the channels, such as the visual, sound, smell, and tactile. Another way of determining the order is based on the sensing ability of the object, whether the object is more sensitive to one channel than other channels. The use of the one-step strategy generally relies on the priorities assigned urgently or personally among the relations.

The two-step strategy can be used to flexibly model the sensing ability of an object in different motion applications. The sensing ability in one channel may also be modeled at different levels to suit the needs of dynamic behaviors. Note the behavior difference which can be easily modeled by changing the order of two channels or two assigned sensing abilities of one channel. For instance, an object modeled with good sight will always show a quick viewing response; an object with an impaired hearing ability will ignore the occurrence of sound.

The *variable structuring property* of relations explicitly describes the changeability of relation control structures modeled among relations in a scene motion. This property is based on the previous three properties (optional, interactive, and selective) and explicitly states that the control structures modeled among a set of relations are dynamically formed from one instant to the next during a motion. The other three properties are examples of this property.

In summary, the optional property shows the subtle behavior differences caused by the influence of other environmental conditions. These differences may lead to differently formed link structures of the relations. The interactive property shows the variable interactive structures of linked relations which can be formed during a motion. The selective property shows the selective use of relations based on one of the two reasons. Each selected set of relations defines an unique basis from which further control structures can be modeled accordingly. From the previous three properties, we obtain the variable structuring property of relations due to subtle behavior differences, variable interactive structures of linked relations, and selective use of relations based on different reasons.

The variable structuring property of relations clearly draws an important conceptual boundary between the motion in a scene and the motion of a single object. In the motion of a single object, the used

control structure is derived from the subparts connection of the object, by either a set of equations or some control constraints. This subparts-structure defined for a single object remains unchanged during the motion. When a scene motion is modeled by relations, the structure among relations dynamically changes while simulating the cooperative, negotiative, and communicative behavior in a dynamic environment. If we compare these two structures (subparts-structure and relations-structure), we find the major difference between the two is static versus dynamic structuring. The relations-structure used for a scene motion does not remain constant during the motion, but dynamically varies as indicated by the variable structuring property of relations. The question of how to explicitly build the dynamic structure of relations is one of the issues investigated in this study.

4.3. Related Objects Mapping

The first step in using the relation control approach is to study the scene problem and decompose it into a set of smaller problems, based on the object-to-object control concept. The solutions to the smaller problems are called relations. Each relation is mapped to two-related objects -- the source object and the responder object.

In dynamic environments, source objects can be both static and dynamic, while responder objects must be dynamic and produce responses. Static objects in an environment include the environment boundary, obstacles, and objects which temporarily appear static in the environment. Dynamic objects include moving objects, scene events, and objects whose properties, such as color, change during a motion. The concept of object is used for both an individual and a group of objects, which have similar properties. Examples of group objects are a school of fish, a flock of birds, and a herd of animals. With the use of this concept, the two object parts of a relation (source and responder) can be mapped to either an individual object or a group of objects.

There are four possible mappings based on whether the source and responder are individuals or groups. These types are one-to-one, one-to-a-group, a-group-to-one, and a-group-to-a-group. The four

mapping types are illustrated in Figure 4.2.

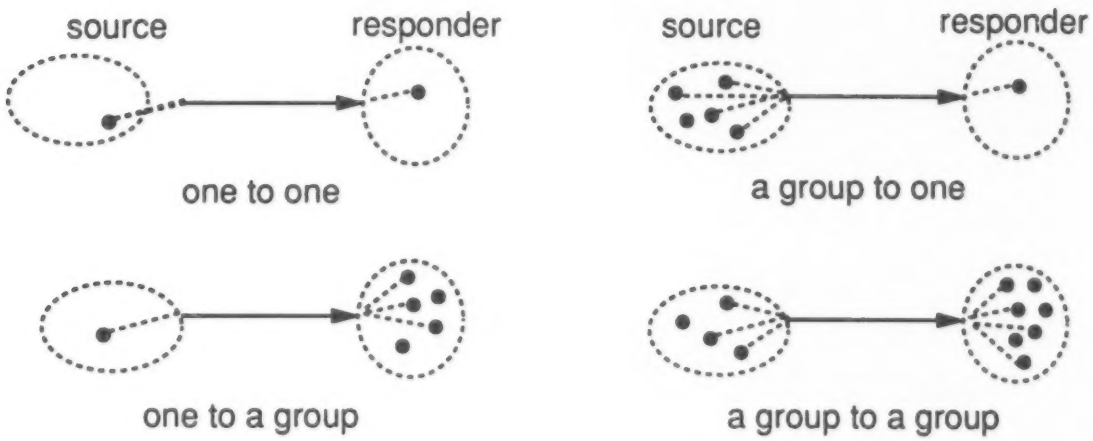


Figure 4.2 Four Mapping Types of a Relation

The *one-to-one* type maps from an individual relation source to an individual responder. Examples of this mapping are relations such as a person opens a door, a dog jumps through a ring, and a fish swims around in a tank. In these examples, the person, dog, and fish are the responders whose motions are enabled by the door, ring, and tank. In this type of relation, one responder is controlled by one individual source. The one-to-one mapping is removed from the control environment if the source object is removed from the scene. If the source object is a moving object or a scene event, the relation may be dynamically switched among the set of relation states depending on how the source changes. For the scene event, the relation may become active when the event occurs in the environment, suspended when the event temporarily disappears, and potential after some period of time.

A-group-to-one type maps a group of relation sources to an individual responder. Examples of this mapping include a shark attacking a school of fish, a dog chasing after balloons, or an actress shaking hands with an audience. In these examples, the school of fish, balloons, and audience are the source group that

influence the individual objects of a shark, dog, and actress. Any member in the source group may become the active source of the relation, which may switch the relation from the potential state to the active state. The active members in the source group can vary from one subset to another during the relation's active state. For the relation of a shark attacking a school of fish, the shark may actively respond to one of the fish in the center of the group or it may be influenced by several fish at the edge of the group.

There are a few ways of simplifying the control of a relation when the source is a group of objects. First, the definition of the relation should be independent of the group size. At the relation description level, a source group should be simply addressed by the group name, similar to an individual. The detailed control on the source group part should be automatically supplied by the relation control system. Second, the members of the source group can be indexed by a variable, which can be varied while modeling a behavior. Third, the summary control over the entire source group can be specified in a separate section of a relation's description.

The *one-to-a-group* type maps an individual relation source to a group of responders. Examples of this mapping are the responses of a school of fish to a piece of food, and a flock of birds flying around an obstacle. In these examples, a group of responders such as a school of fish and a flock of birds are influenced by a single source, such as a piece of food or an obstacle. As with the source group, none of the members in the responder group is active when the relation is in the potential state, and some or all of the members are active when the relation is switched to the active state. The members which are active can vary from one instant to the next. For example, a fish heading away from the food at one moment may not be aware of it, and the birds that have passed the obstacle no longer need to actively respond to it.

In a group response, the response of each group member depends not only on its own sensing ability, but also on the conditions of the other group members. One example is the blocked condition temporarily formed by other group members in front of an object. The source can be missed if other members of the group are currently within the line of sight. The temporary members situation might change the nature of one's sensing. This process may be expensive, as it needs to go through every other group member at every

time interval.

A miss may also occur when a member is moving in the opposite direction to the source, in which case the source is not sensed by the member. On the other hand, the detection of the source by a group member does not necessarily mean that the member can perform the related response. Sometimes the response may be delayed or withdrawn due to the present group conditions. An example occurs when a fish sees a piece of food, but can not approach the food since another fish already has it.

A-group-to-a-group type maps a group of relation sources to a group of responders. The source group can be the same group as the responder group, or a totally unrelated group. Examples of this mapping can be drawn from the schooling behavior of fish, collision among balloons, and games involving two teams. At first, this mapping may be considered an addition to the two mappings of one to a group and a group to one. However, this is not the case. The mapping of a-group-to-a-group can introduce more complex control situations among the group members than the case when there is only one group included in a relation. For instance, if the two parts of a relation are mapped to the same group (such as a school of fish), one member in the group may play both roles of source and responder at the same time. At another time this member may have only the source role, or the responder role. The roles of source and responder can be mapped to various subsets of the group. The process of identifying the member roles of a group may be required at each control interval, as each member role may change over time.

In any environment, the two-parts mapping of relations is not only isolated to individual relations. An object in one relation, either an individual or a group, may be involved in other relations as either a source or responder. An object mapped to multiple relations with the same or different roles is called an overlapped object. Relations can be propagated in a long chain through the overlapped objects. Out of the possible overlapping forms, three patterns are used as primitives to form other complex overlapping patterns. These are the patterns of one-in-one-out, many-in-one-out, and one-in-many-out, based on how one object can be mapped to multiple relations. The many-in-many-out case is formed from the combination of the two patterns of many-in-one-out and one-in-many-out. The overlapping forms of the three primitive

patterns are illustrated in Figure 4.3.

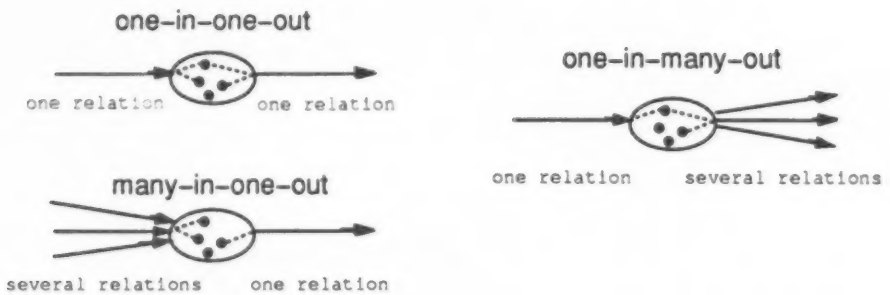


Figure 4.3 Three Primitive Patterns Among Overlapped Relations

The first pattern, *one-in-one-out*, represents the simple overlapping case where one object plays two roles in two relations, the responder role in one and source role in the other. A relation is represented by a linked line which points from the source to the responder. The responder is called the in part and the source the out part. The object in a one-in-one-out role actually plays the responder role in one relation and the source role in the other (in and out here only refer to the representation of a relation). When an object plays interchanging roles, a similar behavior such as the following can be propagated from the first relation to the second. Assuming there is a chain of single overlapped relations in this format, it can be expected that a behavior wave will be propagated from the source object of the first relation to the responder object of the last relation, through all the overlapped objects in between.

The second pattern, *many-in-one-out*, represents the overlapping case where one object plays the responder roles of multiple relations and the source role of another relation. In this pattern, the possible propagated behaviors from multiple relations to the same object can be either integrated from the responses of multiple sources or selected from one of the responses, depending upon the priorities assigned to the relations. One example of this mapping form occurs when a moving object avoids an obstacle, waves to a friend passing by, and attempts to match another moving object's speed. At the same time, the object may

be regarded as an attractive target by another moving object in the environment. In general, if the overlapped object is a group object, the same control principles used for the group members of a relation should be applied to each of the relations.

The third pattern, *one-in-many-out*, occurs when one object plays the responder role in one relation and source roles in multiple relations. As from the same overlapped object, these multiple-out relations can be automatically linked in some structure by the relation control system. Through this structure, the dynamic control information about the shared source object can be directly passed on to the responders. For instance, the relations mapped to a moving object, as the sources, should be terminated when the object moves away from the environment. Another example is a scene event which is mapped as the sources of multiple relations. These relations are accordingly triggered active or inactive upon the appearance or disappearance of the event, such as a series of sound. The event may be caused by an enabling condition as a source.

If the shared source object is a group of objects, the active members mapped to each of the overlapped relations can vary from time to time. At any time, the members of the source group could be involved in all, some, or none of the overlapped relations. Also, different member sets of the source group can be dynamically formed for each of the active overlapping relations. To facilitate the process, some structure for linking the active member set of each relation can be maintained for the relative change of the members.

4.4. Class Types of Relations

To simplify the discussion of individual relations, the diversity of relations in the domain of scene animation can be divided into a few types. This division is based on the type of the source (such as the static source and dynamic source). The common control properties of relations can be discussed based on these few source types. On this basis, the control strategies and mechanisms in dealing with the properties of each type can be introduced.

Relation sources are initially divided into two basic types: static and dynamic. Static sources include objects which present static enabling conditions in the environment, such as the environment boundary, obstacles, and other static objects. Dynamic sources include objects which present variable enabling conditions in the environment. There are two types of dynamic sources: moving objects and scene events. In addition the scene events type can be divided into four event types, which are entering, exiting, updating, and passing.

The *entering event* type includes events which introduce new relation sources into the environment; the *exiting event* type includes events which remove existing relation sources from the environment; the *updating event* type includes events which modify existing relation sources in the environment; and the *passing event* type includes events which bring new relation sources into the environment, but only for a very short period. In summary, the class division of relation types is shown in Figure 4.4. Further discussion of these class types is provided in the following subsections.

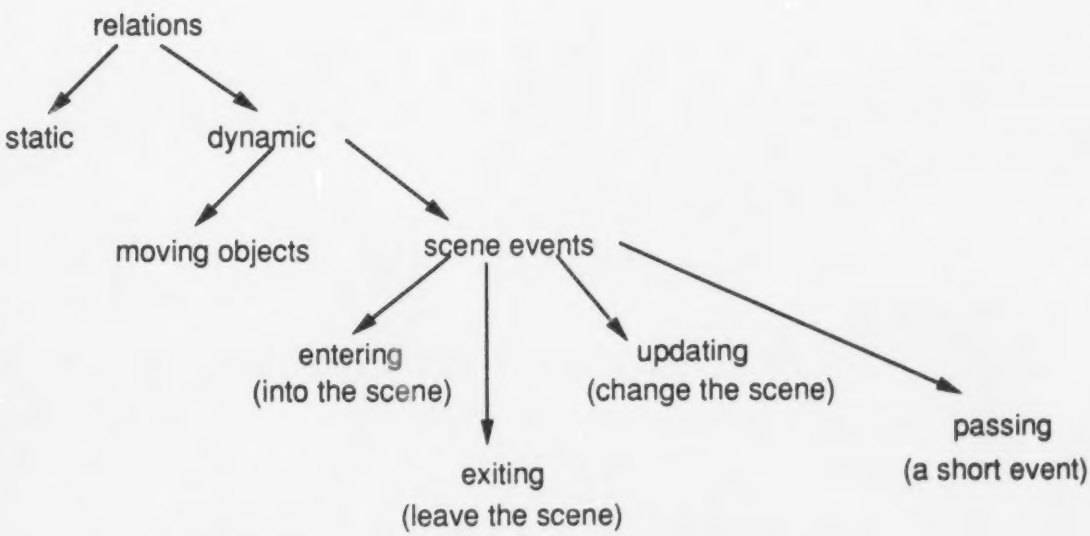


Figure 4.4 Class Division of Relation Types

4.4.1. Static Relation Sources

Relations with static sources are initially selected when the environment is composed. These include the relations for avoiding environment boundaries, avoiding obstacles, bouncing off an obstacle, and approaching a static target. If the environment is modified by adding or deleting various static objects, the relations responding to the updated environment can be partially reselected according to the updated objects. For static sources, both mandatory and optional responses can be modeled. Mandatory responses perform the necessary reaction to the static sources and optional responses perform the secondary reaction or an additional behavior effect.

In a dynamic environment, a static object may temporarily become a moving object and a moving object may temporarily become static. For instance, a static table moves if it is pushed and a rolling ball will eventually stop. These examples show that the initial classification of source type can change during a motion. This change may lead to the use of a different control mechanism.

4.4.2. Dynamic Relation Sources

This relation class covers two types of dynamic sources: moving objects and scene events. Moving objects can cause the value of an enabling condition to change as they move. These changes include the distance from the object to the source, the speed of the source, and certain actions performed by the source. Scene events affect the values of enabling conditions when the events appear and disappear in the environment. Examples of these events include lightning, sound, and a flock of passing birds.

4.4.2.1. Moving Object Sources

If the source object of a relation is a moving object, it can also play the responder role in another relation. When an object plays both the source and responder roles in two different relations, we call these two overlapped relations. One special case of two overlapped relations is the mapping of two objects, where each of them plays one interchanged role (source or responder) in one of the relations. In other words, the source object in the original relation plays the responder role in the other overlapped relation, and vice versa. Thus, one object's response will reflectively depend on the other object's response according to the overlapped relations. This reflective relation mapping can be generally extended to two object groups, which are illustrated in Figure 4.5.

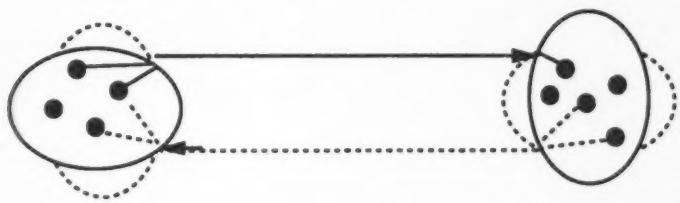


Figure 4.5 Reflective Responses Between Two Object Groups

Two different control mechanisms are used for reflective responses. One is synchronized control and the other is asynchronized control. Synchronized control updates the two objects' responses at the same time based on the conditions sensed from the other object. The previous object states are recorded prior to the control of the two reflected responses, in order for the objects to sense and respond at the same time. Asynchronized control produces each reflected object's response in turn. This means only one object's response is updated at a time. This is supported by keeping a global variable to identify the current responding object. In a mutually reflective scheme, two independent relations between two moving objects can eventually be merged into one relation that both agree with, or to a new relation that both are happy about.

4.4.2.2. Scene Event Sources

The scene event sources can be further divided into four event types, which are: entering, exiting, updating, and passing. The difference between these event types are discussed below.

a) entering events

By definition, these events introduce a new relation source into the motion environment. In response to this newly introduced source, special attention to the control of new relation(s) or existing relation(s) should be addressed. The new relation(s) can include those involved with the new object as either a source or a responder. An example of the former is a moving object avoiding a newly arrived object and an example of the latter is the newly arrived object avoiding a table located in the room.

The new source object introduced by the entering event can be either an individual or a member of a group which existed previously in the environment. A new group member who is the first to enter the environment can be treated the same as an individual. However, for a new group member who is not the first to enter the environment, the event will not introduce a new group relation to the control environment, except for extending the member set. Another change caused by the entering event source may be the introduction of a new relation after some time period. This mainly refers to the type change from dynamic to static. One example of this change is a rolling ball gradually coming to a stop after rolling some distance. Facilities for signaling this type of change in the relation control scheme should be provided.

b) exiting events

When a source object is removed from the environment by an exiting event, the relation or relations involving that source should be deleted from the control environment. The removing control is applied to all relations where the removed source appears as the only responder of the relation. This implies that the relation(s) should be terminated as no one in the environment will be interested in responding to it. However, one exception to termination occurs when the source is a group member and not the last one leaving the environment. In this case, the source rather than the relations should be removed from the source and/or

responder groups.

c) updating events

An updating event can have two effects on a source object: moving the source object to another environmental location or breaking the source object into several pieces at the same location. If the source object is moved to another environmental location by the event, no new relations are introduced to the control environment. However, the type of relations mapped to the source are changed first from static to dynamic, and then from dynamic to static after the move. This may require updating the type control strategies and techniques. If the source is broken into pieces by the event, these pieces can form a source group of the same relation as the original source, or new relations based on the pieces. The use of the same relation or new relations after an *updating* event can be automatically determined by the event handler of the relation control system.

d) passing events

The *passing* event brings a new source object to the environment only for a very short interval. Examples of a passing event are a flock of birds passing over a scene or a lightning bolt appearing in the environment. A passing event can be decomposed into entering and exiting events, where the entering event simulates the appearance of the passing event and the exiting event simulates its disappearance. Under this assumption, the same control strategies used for the entering and exiting events can be applied when a passing event appears and disappears in the environment.

However, a more effective control strategy can be directly derived from two special properties of a passing event. These are the *connective* property and *repeating* property. The connective property is based on the close relationship between the event's appearance and disappearance. This implies that the control of a passing event's disappearance can be performed as an "undo" action on the event's appearance. The repeating property is based on the repetition of the passing event over a limited period. Each time the same updating control of relations, such as adding, updating, and deleting the new relations, is repeatedly applied. To facilitate the repeated control, the new relations introduced by the passing event can be

temporarily switched to the suspended state after the first experience of the passing event. When the event repeats itself, these suspended relations can be quickly resumed to the active responses; if not, these suspended relations should be released after an idle period.

4.5. Minimum and Maximum Number of Relations

The control complexity of a scene animation can be measured by the number of relations used in the environment. The more relations that are used, the greater is the control complexity among these relations. The use of relations can be predicted from the objects used in the environment. Each relation includes two related objects, the source and responder. One example of a relation mapping graph for an environment with three static objects, two dynamic objects, and a sound event is presented in Figure 4.6. For clarity, the static objects in the graph are drawn using solid lines, the dynamic objects are drawn using boldface lines, and the sound event is drawn in the dotted line. In this figure, only one relation is mapped between two objects.

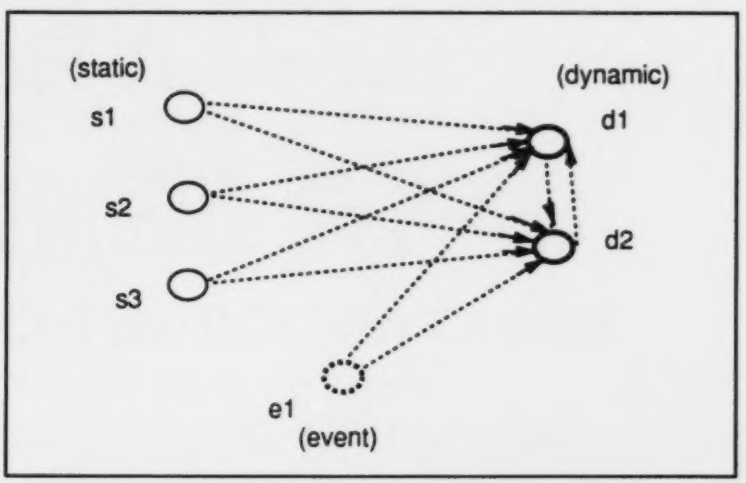


Figure 4.6 Example of Relation Mapping Graph in an Environment

The simplest control in a scene animation occurs when the least number of relations are used in the animation. This occurs when the only motion in the scene is obstacle avoidance. In this case, only the avoid relations are used between moving objects and other objects in the environment. In general, if an environment is composed of n static objects, m dynamic objects, and the environment boundary as one entity, the minimum number of relations for avoiding other objects in the environment is

$$R_{minimum} = m*n + m*(m-1) + m$$
$$= m*(n+m)$$

In this equation, the first item represents the relations involving the n static objects and the m dynamic objects, the second item represents the relations among the m dynamic objects, and the third item represents the relations from the environment boundary to the m dynamic objects.

The relations used for obstacle avoidance in an environment are far from the relations needed to produce interesting, individual, and personal motion around an environment. There are many other alternative relations which can be used to produce interesting motion behavior. Even for the necessary avoiding relations, there may be more than one way to avoid obstacles. These alternative relations can be derived from the experience of a moving object, personal preference of each individual, other dynamic influences at the avoiding time, as well as additional responses to the scene events.

The alternative relations based on the experience of a moving object are differentiated by the number of times the object responds to the same related source. Without knowing the exact number of times alternative relations are used (which may depend on each application), the maximum number of alternative relations for modeling an object's experience can be estimated to a t_i factor, whose value is in proportion to the maximum number of times an object responds to the same related source.

Alternative relations used to demonstrate an individual's character are differentiated by the changes of a character during a motion. The maximum alternative relations for modeling a character's change can be estimated to a p_j factor, whose value is in proportion to the number of characters modeled for an object.

The alternative relations for the additional responses to scene events are differentiated by the events occurrence in the environment. Thus, the maximum alternative relations can be estimated by $m*k$, where m is the number of dynamic objects and k is the number of scene events present in the environment. The alternative relations for the reflective responses of two moving objects can be estimated to a d_l factor, whose value is in proportion to the number of reflected responses between the two moving objects.

By counting these variable factors, we use additional coefficients representing the alternative relations for each of the terms derived in the minimum relation equation. The original equation is also extended to a new term for the relations involving scene event sources. The extended equation represents the maximum number of relations which can be used in a general environment context. In this extended equation, the coefficient C_s, added to the term involving static object sources, represents the average number of enabling conditions modeled for a static object. The coefficient C_d, added to the term involving dynamic object sources, represents the average number of enabling conditions modeled for a dynamic object. The coefficient C_b, added to the term involving the environment boundary source, represents the number of possible enabling conditions modeled from that boundary. The coefficient C_e, added to the new term involving dynamic scene event sources, represents the average number of enabling conditions modeled from such a source. In summary, the maximum number of relations when using alternative relations in the modeling of a scene motion is

$$\begin{aligned} R_{alternative} &= C_s*m*n + C_d*m*(m\text{-}1) + C_b*m + C_e*m*k \\ &= m*(C_s*n + C_d*(m\text{-}1) + C_b + C_e*k) \end{aligned}$$

The above analysis for using alternative relations provides a basis for comparing the control complexity of environmental behaviors. The coefficients counted for each relation source type can be increased from the minimal configuration of relations, which should be first conducted in a given environment, to the maximum use of alternative relations.

Estimating the number of relations possibly used for a scene motion application also outlines the basic complexity layer for modeling and controlling such an application. In general, if more relations are

used, then a more complicated control situation can be presented. This potential increase of relation control complexity is due to the fact that relations do not behave independently. Instead, their dynamic behaviors can interfere with each other in a rather complex and even unpredictable pattern during a motion. The estimated number of relations can be similarly counted for the time complexity for running such an animation.

Chapter 5

Outline of the Relation Model

This chapter presents the technical "skeleton" for using the relation control model. The "skeleton" contains two parts: relation modeling and relation structuring. Relation modeling uses a general frame format to describe the local control properties and the behavior of each relation. A relation only performs a simple responsive behavior between the two related objects. To model the complex, dynamic behavior in an environment context, relations are structured into a hierarchy. Four structuring levels are derived in the hierarchy, for addressing the important behavior control aspects of the animation. These four levels are: selective control, state control, pattern control, and sequential control. A general algorithm for processing the relations at each control step of the animation is outlined, along with its time complexity analysis.

5.1. Frame Modeling for Relations

The relation control frame is used for modeling relations of different types. Common control properties of relations, such as the source and responder objects, enabling condition, sensory channel, response strength, response duration, and response behavior, are included in the relation frame description. The relation frame is divided into three sections: *control header*, *action code*, and *finalization*, as shown in Figure 5.1.

The *control header* of a relation frame consists of a set of control slots, whose initial values can be specified at relation declaration time. The control slots in the relation control header are: initial sensing state, effective duration, names of the source and responder objects, enabling condition, selective priority, response strength, optional switches, and motion aspect (used for possible conflict prevention). The use of each of these slots is now described.

The *initial sensing* slot is used to specify whether a relation can initially sense its enabling condition. If a relation is specified as initially active, with the keyword ACTIVE for the slot value, its enabling condition can be actively sensed at each control step. The initial active assignment at this slot signals the

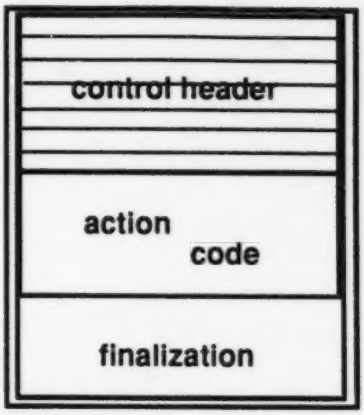


Figure 5.1 Three Control Sections of Relation Frame

potential state of a relation. Otherwise, the relation is in the suspended state. The default value for this slot is inactive sensing (suspended state). If a relation is not initially in its active sensing (potential state), it can still be changed to this state by other active relations or modeled behavior structures.

The *effective duration* slot is used to specify the time period that a relation will be active once it is triggered. A relation will terminate its response when this duration is over, or when its enabling condition becomes false. The normal response duration, however, can be interrupted by interactions amongst relations. Such an interruption can either delay the active response for some time or completely terminate the response. An interruption can be caused by an inter-relation call, other structured behavior control, or a priority-based selection. Also, the initial duration value can be varied through the interactive interface of a relation control system.

The *name* slots are used to state the source and responder objects. The source and responder objects will be the same object, if one of them is not explicitly specified. Both source and responder can be either an individual object or a group of objects. The mechanism for mapping a relation to a group object is automatically handled by the system. The difference between naming a group of objects and an individual

group member requires special attention. The group case is specified by the group name, and the member case is identified with the particular group member index, such as "object_name(i)" for the *i*th group member.

The *enabling condition* slot describes how the source is sensed by the responder. The condition sensed from the source can be specified as "channel name: enabling condition". In this format, the channel name indicates one of the standard sensing channels (visual, sound, smell, and tactile) and the enabling condition indicates the conditions that must be satisfied to enable a response. For example, the distance condition in the visual channel can be specified as "visual channel: ifclosetoblock(distance)". The condition becomes true when the distance to the source object falls within a given threshold, which is provided by a parameter. If the condition is true, the reaction defined in the relation body is performed. The enabling condition for the other standard channels, such as sound, smell, and tactile, are specified in a similar way.

The *selective priority* slot is used to assign a competitive priority among relations that could conflict or produce alternative responses. There are two types of priority assignments: urgent and personal. The urgent priority assignment is based on the relative order of importance amongst the relations, and the personal assignment is based on the personal preferences of individual characters. Priorities assigned for different reasons can be separated into value ranges, with the highest value having the highest priority.

The *response strength* slot contains all the parameters that control the response process of the relation. Three types of response parameter values can be specified, which are the direct type, functional type, and random type. The direct parameters passes the assigned parameter value to the process directly. The functional parameters first applies a function, such as linear interpolation, to the parameter before it is passed to the relation. The random parameters pass a random value within a specified range to the relation. Consequently, the use of these parameter types can achieve either a definite, functionally variable, or random control effect when the object responds to the environment.

The *optional switches* slot is used to specify subtle control over the object's response based on other objects in the environment. These could be nearby objects or an object currently passing by. Depending

upon the presence of other environmental conditions, an object could behave slightly differently towards its related source, in addition to its normal reaction. This subtle difference is modeled using switches. Once a switch variable is specified in the switch slot, its value can be interactively switched to "on" or "off". Each interaction with the switch reverses its state. With the "on" value, the subtle modification of a reaction is enforced. Otherwise, no change is made to the normal reaction. One example of a subtle modification of a relation is to change the direction of a turn or extend the duration of a look, when another object in the scene moves closer.

The *motion aspect* slot is used to specify the part of the motion that the relation controls, such as the movement of an arm or a leg of a human figure model. More detailed description can be the name of a force or a torque used at a body's joint to generate the motion, such as a push at the left_elbow. Based on the explicitly specified control aspect, possible conflict amongst the active relations can be easily predicted. This quick prediction can speed up the control process and guide the proper use of mutually exclusive relations. If a possible conflict is detected for the same control aspect, the conflict is avoided by selecting the relation with the highest assigned priority.

The *action code* of a relation frame describes the relation's response. The action code can be specified by a script, a procedural notation, a high-level animation language, or a natural language, as long as the descriptive tool used has sufficient power to express the reaction of the object. For instance, if a procedural notation is used for the action code, all the control facilities, including conditional test, looping structure, and subroutine call, defined in the procedural notation can be used without extra cost.

In addition to the use of one existing language, other rules for directly citing the slot names and expressions from the control header section are also included in the action code. One of them is the channel name reference, which refers to the enabling condition specified in the channel slot of the control header. This reference retrieves the condition sensed by the channel, without stating the condition directly in the code. This simplified reference is simpler and provides clearer description of the problem. One example of a visual channel reference is "visual channel: statement | {statements}". Excitatory and inhibitory

responses based on a channel's current state can be simply expressed as "channel name" and "!(channel name)". The AND and OR operators can be used to combine the responses from several channels. In this type of expression AND has higher priority than OR unless parentheses are used.

One initial motivation for having the action body of the relation frame is the ability to use different types of motion control techniques. These include the commonly used techniques such as kinematics, dynamics, stochastic process, and 3-D parametric keyframes. The selective use of these techniques depends on the application requirements. It also depends on the range of motions which can be modeled with the available computing power and special motion effects. For instance, the use of kinematics leads to a quick and reasonably realistic motion in simple problem domains. The use of dynamics produces more realistic motion in a fairly complex domain with a relatively slow control process. Also, 3-D parametric keyframes provides a general control mechanism for a broad range of motions, based on the similar keyframing technique used in the traditional animation.

Since there is more than one control technique which can be used in animation, we would like to keep this option open in the definition of the action code. The description of the action code of a relation should not be restricted to the use of one control technique. It should be the user's choice to decide which technique to use according to the application's need and the user's experience. For this reason, rules defined in the action code are fairly general, abstract, and high-level. The use of these rules can be easily adopted to other control techniques. Examples of the current system version, shown in Chapter 6, are produced using procedural control of 3-D parametric keyframes, using the C programming language as its host language.

As stated before, two object forms are used in modeling relations, an individual and a group. For both forms, only the object name is used in the relation description body. A relation is described using only the names of the source object and responder object, where only the object types matter in the description of a relation. From this reason, we can view group behavior as an extension of a single object behavior to the whole group. It is the system's responsibility to check for the case where the named object stands for an individual or a group and how the group control facilities are applied. To do that, the system uses its

previous knowledge about the named object from an earlier object declaration and applies the proper group facilities while interpreting the code described in the action body of a relation.

One exception to the general group extension rule is the modeling of an individual group member's behavior. If a member's behavior is completely different from the rest of the group, a separate relation addressing that member can be produced. This relation is specified using the member indexing function in either the source or responder slot. If a member's behavior is only slightly different from the others, this slight difference can be addressed using the member indexing function in the action code of the group relation. By changing the parameter of the member indexing function, either a specially modeled relation or a slightly varied behavior in a group can be reassigned to another member. Slight behavior variation of several group members can be similarly treated using two parameters to define an object subset that receives special treatment.

Communication among relations can be modeled in the action code using one of the state control functions, which are state_active, state_blocking, state_deactive, state_suspended, and state_terminate. The C procedure declarations for these procedures are:

```
void CallActive(relation)
char *relation;

void CallActiveBlock(relation)
char *relation;

void CallDeactive(relation)
char *relation;

void CallSuspended(relation,interval)
char *relation;
int  interval;

void CallTerminated(relation)
char *relation;
```

The CallActive(relation) routine changes the state of the named relation to the active state, which triggers that relation's response, if its enabling condition is currently true. The CallActiveBlock(relation) routine changes the state of the named relation to the active state and at the same time moves the relation issuing

the call from the active state to the suspended one. The relation is suspended during the active response of the relation it calls. The CallDeactive(relation) routine changes the state of the named relation to the potential state, and this relation can become active as soon as its enabling condition is true. The CallSuspended(relation,interval) routine changes the state of the named relation to the suspended state for the specified time interval. Once the interval is over, the relation's state returns to its original value. The CallTerminated(relation) routine changes the state of the named relation to the terminated state.

Other state controls are used for relations that are scheduled with respect to a referenced relation. These relations can become active at any time during the active period of the referenced relation. To check for this possibility, a relation can issue three function calls during its active response. These calls check for the relations occurring at the three time references, and issue an active-call if such a relation is found. These three times are: initial active time, in-between active time, and final active time. Relations scheduled at one of these times will be called active by the routine issued at that time. Three checking routines for the three referenced times are declared in C as:

```
void Call_initial_active()
void Call_in_active()
void Call_final_active()
```

Among them, Call_initial_active() checks for relations scheduled at the initial active time of the issuing relation. Call_in_active() checks for relations scheduled at some time during the active duration of the issuing relation. Call_final_active() checks for relations scheduled at the final active time of the issuing relation. These controls can be automatically checked by the system, for each active relation at each control step. No explicit routine statements are necessary in this case. To reduce unnecessary search, our current system uses explicit statements for this task, where only the relations with the routine statements are checked.

Two other routines used for changing a relation's state are Set_active() and Set_deactive(). The routine Set_active() sets the state of the relation issuing the call to active, and routine Set_deactive() sets the state of the relation to potential. The C procedure declarations for these two procedures are:

```
void Set_active()
void Set_deactive()
```

These two routines are issued at either the initial active time or the final active time of a relation. When such a state transition occurs, either from the potential to active or from the active to potential, these routines change the relation's state in the system's relation table. Again, these self-state controls can be automatically checked by the system, while observing the dynamic state transitions of each relation at each control step. Currently, it is left as part of the state control facilities issued inside a relation's body.

The *finalization* part of a relation frame is used to specify the summary control that is necessary at the end of a response. When a relation involves a group, either a source group or a responder group, such a control over the group is normally required. The control could be the computation of an average response strength, selection of an extreme reaction, and an ending message from a group response to other relations. Other possible actions at the end of a response could be self-state control, interactive state control, structured control, and the control necessary at the end of a response. To meet these needs, the finalization section of a relation frame provides a separate place for specifying these actions. This place separated from the action body section is only processed at the end of a response, which can be a response extended to a group object.

The general format of a relation frame is shown in Figure 5.2, where the C procedural description is simply included within the pairs of curly braces together with the other relation statements.

```
/*    control header of relation frame
source name: string      /* either an individual or a group
responder name: string   /* either an individual or a group
channel name: Boolean expression  /* enabling condition(s)
  . . .          /* other sensing channel
initial state: constant
effective duration: integer
response parameters:
   parameter declarations  /*  see below
switches: string [On | OFF]
priority or preference: integer
motion aspect: constant      /* position, orientation, color, etc

 /*  action code of relation frame
```

```
relation name  {
    . . .    C statements -- required before conditional test
  channel name: {
          . . .
             C procedure calls -- for state control
          . . .
             C statements -- for computing the excitory responsive
                             behavior
          switch: {
            . . .
             C statements -- for computing subtle response difference
                             upon the switched condition
               }
            . . .
             library references -- for common response control
            . . .
          !(channel name){
            . . .
             C statements -- for computing the inhibitory responsive
                             behavior
           }
        }
   (channel name AND .. OR .. ):  {
          . . .
             C statements -- for computing the response based on the combined
                             conditions sensed from multiple channels
          }
          . . .
    }

  /*     finalization of relation frame
==> {
        . . .
        . . .
             C procedure calls -- for state control
             C statements  -- for finalization control
    }


    /*
    /* the response parameters have one of the following forms
  string number              /* control type
  string number1 -> number2    /* dynamic type
  string number1 <-> number2   /* variation type
```

Figure 5.2 Simplified Version of Relation Control Frame

5.2. Hierarchical Structuring Mechanisms

There is no question that using a hierarchy is a good way of organizing massive amounts of information into a more understandable and effective form. This idea has been used in the area of computer animation, since the amount of control information for many motions tends to be large. The hierarchies used in animation are based on the levels of control details for the motion. The higher the level, the more abstract the control information that is present. For instance, in most animation hierarchies, the task level is the highest level at which a motion can be specified, either by natural human language or system commands such as "walk from here to there" or "sit down on this chair". This high-level interface is converted into more detailed control information, such as the walking cycle, and arm and leg movements in the cycle, which form the middle levels of the hierarchy. At the lowest level each position and orientation for the geometrical primitives must be produced.

Hierarchical structures based on the amount of control details also represents the levels of the user's understanding of the motion. With different levels of training and motion control knowledge, users can selectively work at one suitable control level. Users with little knowledge of a motion can learn from the experience of viewing the motion issued by a natural statement or a command at the highest hierarchical level. Users with more motion control knowledge can learn from the experience of interacting with the motion at various middle levels. And, users with sufficient control knowledge and skills can learn from the experience of working at the lowest level, which gives the maximum control over the motion.

Another advantage of using a control hierarchy in animation is the control structures supplied through the hierarchical levels. These structures, from the lower levels to the higher levels, offer direct control of related motion aspects. It gives a great benefit for the user to trace controls related to a single aspect, such as the movement of the left arm. The task of tracing the levels of control in a linear program can be very difficult and time consuming. A hierarchical structure is one way of getting around this problem. With the explicit structure of a motion, the time and effort required to produce a desired effect is greatly reduced by the structural guidance. Also, the motion can be modeled, tested, and modified in a direct and effective

manner.

The same reasons motivate the use of a hierarchy in our motion control model. We use a hierarchy to organize the large amount of control information required for animation, and to permit the system to adapt to users with different knowledge, skills, and purposes. We also use the hierarchical structures to separate and organize the related controls of each behavior aspect. However, as relations are used as the basis for building such a hierarchy, the goal and mechanism for using a hierarchy in the relation control scheme is quite different from the traditional approaches.

One of the differences with the traditional approaches is the way a hierarchy is used. Unlike other hierarchies built on the concept of movement, from more detail to more abstract, our hierarchy uses the concept of relations. Our hierarchy organizes information based on how two objects are related to each other, and how one object is related to the environment. If a relation's condition is true, a response is automatically triggered by the relation. In this hierarchy, the lowest level deals with relations, modeled as primitive control elements, rather than raw geometric information. Another difference is the purpose of each hierarchical level. The levels in our hierarchy are not merely used for the purpose of forming a complete motion, but for adding other related controls of a motion in the environment. A motion can be issued at each level of our hierarchy, not just the top level. When motion is structured through the levels, more responsive, interesting, and realistic motion in a dynamic environment is produced.

Because of these two essential differences, our relation hierarchy can be used for a wide range of behavior applications in an environment, rather than just one motion application. The application scope of our hierarchy extends from a predefined behavior sequence to a set of behaviors constructed from a set of relations modeled at the lowest level. Our hierarchy also provides the user a learning environment for incrementally modeling a complex and dynamic behavior from simple relation behaviors, by working up through the hierarchical levels.

In the following subsections, previous control hierarchies that have been used in animation are introduced. Following that, our control hierarchy for behavioral animation is outlined in detail.

5.2.1. Previous Research on Control Hierarchies

Zeltzer [Zeltzer85] has identified three hierarchical control modes based on a survey of a number of character animation systems. These three control modes are called guiding systems, animator_level systems, and task_level systems. The levels of control modes are based on the control approaches used in the character animation systems. The guiding mode covers the systems with no mechanisms for user-defined abstraction. Thus, motion control can not interact with the environment and no feedback information from the control process is used by these systems. The animator_level systems allow the animators to specify motion algorithmically. Examples of these systems include abstract programming languages and message passing systems. The task_level systems give the animator facile control over complex motions by trading off explicit control over the details of motion. Zeltzer claims that a promising approach is to integrate these three control modes together into one animation system, so that the weaknesses of one control mode can be overcome by the use of other modes.

The skeleton animation system designed by Zeltzer [Zeltzer82] uses a three level control hierarchy for human figure motion. This hierarchy consists of the three levels of task control, skill control, and primitive control. Among them, the task control level is the highest one, accepting a task description from the user and decomposing the task description into a list of skill components, such as walking, running, or grasping. These skills are passed to the skill control level, in which they invoke a fixed set of primitive procedures which are parameterized. A primitive procedure, at the primitive control level, can access a fixed list of joints in the human figure model and change the current rotation values of these joints in the system's skeleton database. Through the levels of this hierarchy, a figure's motion is structured into a regular motion sequence of primitives. The parameters at both the skill and primitive control levels can be varied within certain ranges. The level division of the control hierarchy used in the skeleton animation system is outlined in Figure 5.3.

A four level hierarchy for what an object is has been proposed by Barr [Barr89]. In this hierarchy, the first level, called "object-as-image", represents two dimensional modeling primitives, primarily pixel

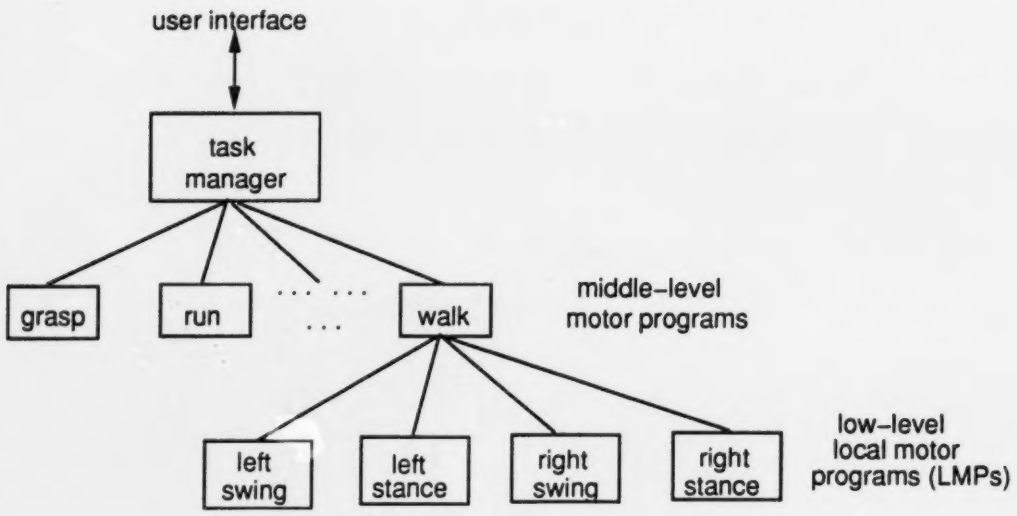


Figure 5.3 Levels of Control Hierarchy in the Skeleton Animation System

images and vector line drawings. The next level, called "object-as-shape", represents three dimensional kinematic primitives, including polygons, patches, and the like. The third level, called "object-as-behavior", represents objects as rigid and flexible physical bodies. And, the final level, called "object-as-timeline", incorporates time-dependent goals of behavior or purpose as the fundamental representation of what the object is. Using this approach, objects can be modeled at different levels of abstraction, including the modeling of objects generated by an animation. The most interesting idea about this hierarchy is the clear outline of a new graphics pipeline. Through the pipeline (levels of the hierarchy), objects can be modeled using different representations from one level to the next. The levels proposed in Barr's hierarchy are outlined in Figure 5.4.

A hierarchical reasoning system called HIRES [Badler86], permits modeling of an activity at multiple levels in different representations. The levels of the system are based on continuous system simulation models, discrete simulation models, petri nets, timed petri nets, and scripts. To simulate a motion, the sys-

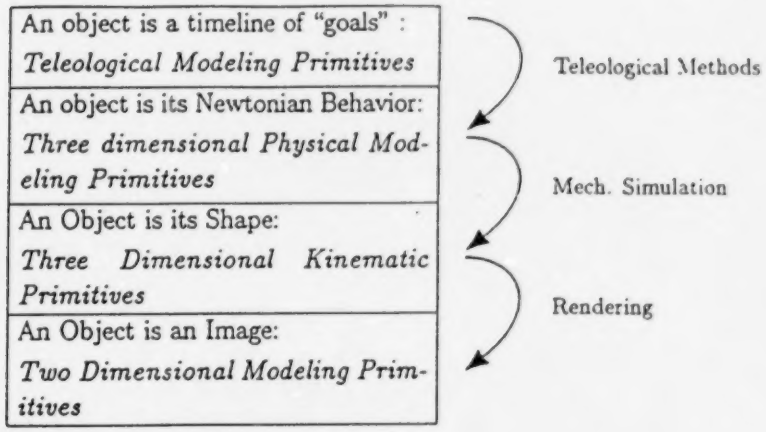

An object is a timeline of "goals" :
Teleological Modeling Primitives
An object is its Newtonian Behavior:
Three dimensional Physical Modeling Primitives
An Object is its Shape:
Three Dimensional Kinematic Primitives
An Object is an Image:
Two Dimensional Modeling Primitives
Teleological Methods
Mech. Simulation
Rendering

Figure 5.4 Levels of the Barr's Hierarchy

tem supports the representation of a dynamics problem using a continuous simulation paradigm at the lowest level. The next level might be supported by a queuing system (discrete) model, including parametric keyframes, while the upper levels can be symbolic, state-based, qualitative-reasoning models. If these descriptive levels are appropriate for the animation process being controlled, the problem of animating natural, fluid, coordinated, task-oriented, and expressive motions can be expected as the system uses the power of other research areas extended into the graphics domain.

5.2.2. Relation Composition Hierarchy

The control hierarchy proposed in this thesis is based on relations. Prior to structuring relations into a hierarchy, relations are modeled as primitive controls. The modeling of relations should be as simple as possible. In the previous section, we have described how relations are modeled using the relation frame. Here, we show how relations are structured into hierarchies in order to produce more complex behaviors.

Since the control hierarchy proposed by this research is quite different from the hierarchies used in other animation applications, two special environmental properties should be addressed by the hierarchy. One is the flexible control of responses to dynamic changes in the environment, and the other is the change

of environmental behaviors. We expect that the motion built by a hierarchy can be easily modified in response to changes in the environment, that can happen during an animated sequence. Such changes include the appearance or disappearance of an object, a different initial environmental setting, a delayed event occurring, or an object added, replaced, or deleted from the environment. The structures built in the hierarchy should support such a change, similar to the change of an environment. Here a movement composed of relations resembles an environment composed of objects.

The control of relations that can form changeable behaviors is another special feature of our hierarchy. Using a hierarchy, we can structure relations to describe an individual, desirable, and personal environmental behavior. To this end, effective control mechanisms are required to address the transition from a set of relations to an environmental behavior. The transition is more complex than a collection of relations, each behaving individually, or a simple addition of the relations. Instead, many unpredictable control patterns of relations can be dynamically formed in both the time and space dimensions during a motion. The structures built in the hierarchy explicitly and effectively address these dynamic processes.

In addition, the use of a hierarchy will facilitate changing behaviors. A behavior once modeled through the hierarchy can easily be modified to produce a slightly different behavior. Thus, the hierarchy not only facilitates the modeling of one behavior, but also a set of similar behaviors each ruled by a different character. The use of a hierarchy can provide a flexible testbed for behaviors, where a set of behaviors can be easily explored in terms of the structures built for the motion. The hierarchical transition from a set of relations to an unique behavior can be extended to the transition from a set of relations to a set of similar behaviors.

In summary, a control hierarchy based on a set of relations can play an important role in explicitly and effectively organizing, modeling, and editing one or a set of similar behaviors, which are independent of environments, personal characters, and purposes.

We propose a hierarchy with four relation composing levels, called *selective control*, *state control*, *pattern control*, and *sequential control*. This hierarchy is based on the complexity of an environmental

behavior, growing from a simple, carefree behavior to mature, responsive behavior. Selective control composes the set of relations that are necessary for the current environment and behavior. State control composes the state transitions among selected relations. Pattern control composes a set of relations to produce identifiable behavioral elements. Sequential control composes behavior patterns from the lower levels to produce ordered sequential behaviors.

One major difference between our hierarchy and other proposed hierarchies is the flexible and versatile use of a hierarchy. Our hierarchy is not limited to one complete motion that is issued at the highest control level, as other hierarchies are. In our case, each hierarchical level expands an object's behavior to a more dynamic, responsive environmental behavior. Motion with the behavior modeled at each hierarchical level can be independently tested as the levels are constructed before the entire hierarchy is constructed.

The levels of our relation hierarchy are outlined in Figure 5.5, and more detailed discussions of these hierarchical levels are presented in the following subsections.

5.2.2.1. Selective Control

The input to the selective control level is the relations that have been modeled using the relation frame description. Each of these relations depends on two objects for its existence: the source and responder. When a relation is added to the system, its two objects as well as its other local control properties are recorded in the relation table, while its main response and finalization code are copied to the relation library. With this information, the relations that can be used in an environment, based on the objects in the environment, can be determined. A relation is selected by an environment if both its source and responder objects are present in the environment. If the environment is changed by adding or deleting one or more objects, the set of selected relations will change based on the objects changed in the environment. This environmental selection of relations can be automated, by examining each relation to see if both of its source and responder objects are included in the environment. This search can be repeated when the environment is changed.

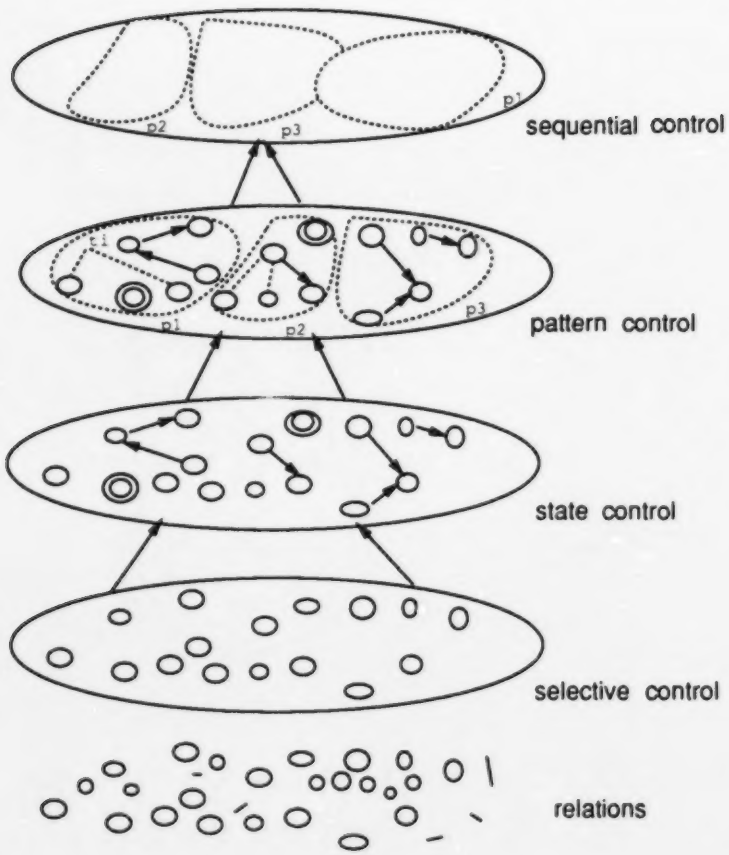


Figure 5.5 Levels of the Relation Composing Hierarchy

The second selection of relations is based on the modeling of a behavior. We call this selection behavioral selection. An environment selects all the relations which are able to react in the environment, and a behavior selects the relations which are necessary for modeling the behavior. In the current system, this control option is left to the user who can interact with the relations menu of the system. The user suggests the previously selected relations that will be used in the current application, using a menu.

This second selection option allows the user to experiment with several behaviors in the same environment. It selects the relations that are used as basic elements in the higher levels to form a desired

behavior. Users with different taste, experience, and purpose can choose to reselect the relations that are used in the environment while modeling a behavior. One example of behavioral selection is the selective use of relations in response to a sound. When a sound event is added to the environment, relations using the sound source are initially selected by the environment. These initially selected relations, however, can be selected again while modeling individual behaviors, such as a quiet response at the first sound appearance, but disturbed responses to subsequent events. This behavioral selection option can also be automated, by using the behavior knowledge stored in the system.

5.2.2.2. State Control

State control is used to control the state transitions of the relations selected in the previous level. A relation can only respond when it is in the active state. The other three states describe either potential, suspended, or terminated status for the relation. Initially, a selected relation can be in one of the two states, potential and suspended. A relation is in the potential state if it is initially assigned the active sensing state. A relation is in the suspended state if it is not assigned an initial state. During a motion's progress in the environment, relations selected in one of the two initial states are possibly converted to one of the other states. Whether a relation is converted or not depends on the relation's sensing, duration, and interactions with other relations.

There are many reasons for modeling the interactions among relations, that leads to state transition in a relation. Some of the reasons are mutually exclusive, competitive, and cooperative behaviors. Other reasons are personal taste, preference, experience, and motivation of an individual. The structures used at this level to model state transitions are: self state control, interactive state control, and environment state control. Self state control is due to the enabling condition assigned to a relation. A true enabling condition occurring when the relation is in the potential state will lead to a transition to the active state. The active state changes to the potential state when the enabling condition becomes false, or the response duration is over. Only an active enabling condition for a relation (potential state) can lead to an active response (active

state) by itself.

Interactive state control occurs when a relation's state is changed by other relations. This control is issued from an active relation to another relation that can be active or inactive. Five state controls can be issued by an active relation (see Section 5.1 for a relation's description):

CallActive(relation)
CallActiveBlock(relation)
CallDeactive(relation)
CallSuspended(relation,interval)
CallTerminated(relation)

Among these calls, only the CallActiveBlock(relation) is an inter-dependent control between two relations (calling and called relations). The calling relation, the one issuing the call, blocks itself during the active duration of the called relation. This control can describe mutually exclusive use of two relations. This interaction produces a delayed response relative to another active relation, such as the relation in response to an event currently occurring in the environment. Other calls issued by the calling relation only change the state of the called relation, not the state of the calling relation.

Environmental state control describes the state changes that occur due to events in the environment. The composition of the environment determines the first selection of relations either in the potential or suspended state. The dynamic object sources in the environment also affect the state transitions of relations. For instance, the time and length of an event in an environment can trigger state transitions among relations. The dynamic behavior of a moving object can be modeled using state transitions. Examples are the use of a newly active relation and a delayed relation whose behavior is temporarily blocked when an event occurs. In general, when a dynamic source such as an event or a moving object appears or disappears in an environment, relations influenced by the dynamic source are possibly triggered from one state to another. In this case, we say that the composition of a dynamic environment is the basis of possible state transitions.

The three structures leading to a possible state transition are illustrated in Figure 5.6. In this illustration, a double circle represents a relation where the inside circle shows the response and outside circle

shows the condition. The type of color used to draw the circles denotes the current state of the relation. A red-colored outside circle indicates the active enabling condition (potential state), a red-colored inside circle indicates the active response (active state), and both circles in green show the suspended state. The relation(s) currently in the terminated state should be instantly removed from the relation control diagram. An interactive state control is drawn as a line from the interacting relation to the interacted relation, where a letter along the line indicates the changed state of the interacted relation, such as letter "a" for the active state, "p" the potential state, "s" the suspended state, and "t" the terminated state.

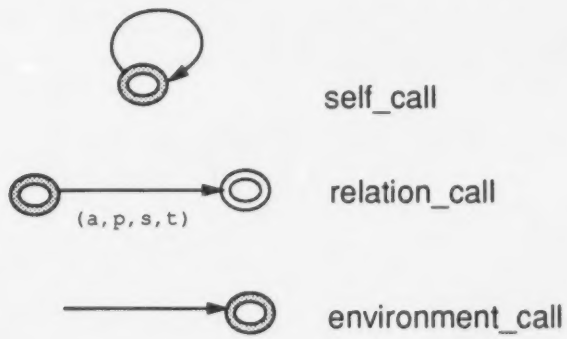


Figure 5.6 Three Structures for a Possible State Transition

5.2.2.3. Pattern Control

At the pattern control level, relations are combined into various behavior patterns. These patterns are formed by the grouping, scheduling, and recording control mechanisms. Each of the patterns produces a complex behavior unique either to a local environmental situation or to the local time reference modeled in the pattern. The environmental reference stimulates the active set of relations and the way to structure the relations. Special behavior can be modeled when the situation occurs. The time reference in a pattern adds other subtle responses relative to the main action stream of the pattern. The pattern modeled by this reference can be commonly used in various environmental situations.

A grouping structure causes a set of relations to be active at a time relative to the beginning of a behavior pattern. A grouping structure places the grouped relations in the potential state at the referenced time, and they can change to the active state when their enabling condition becomes true. Whether these relations will take an active role in the motion depends on the value of their enabling conditions. For the relations modeled with the default true enabling condition, a case used for common relations, the grouping structure directly brings these relations to their active responses (the active state). In this case, the active sensing state is immediately transferred to the active response state.

If the relation is in the suspended state, it will immediately recover to the active state once it is called by a grouping structure. If the relation is in the potential or active state, no difference is made by the call issued to the relation. If the relation is in the terminated state, an error is reported.

The scheduling structure causes a set of relations to be active when the referenced relation changes to the active state. Three time references can be used, which are: the first time step, the last time step, or one of the time steps during the active response. When the referenced relation becomes active, it checks for the possibility that a group has been scheduled at its initial active time. If there is one, the relation calls the scheduled group as one of its initial activities. Checking for the relations scheduled at the other two time references is similarly processed. A group scheduled during the active time is checked at each time step of the motion, and groups scheduled at the final time step are only checked prior to the time when the referenced relation ends its active response. A schedule can also reference a scene event, at the time that the event occurs, continues, or ends.

Recording is the third mechanism used for modeling behavior patterns. The relations in the lower levels and the same control level (grouping and scheduling), are recorded and given a name. The recording only copies the necessary control structures built among the relations, including their states, local control properties, interactive controls, and grouping and scheduling structures. The copy is limited to the relations which are used in the pattern. The recording provides a better control interface for the next level of the hierarchy, where the named behavior patterns can be directly referenced as elements while modeling a

sequential behavior or behaviors.

5.2.2.4. Sequential Control

Two sequential control mechanisms are used, which are scaling and ordering. Scaling controls the length of time that a pattern is active, which can be either stretched or compressed in the time dimension. If a behavior pattern is selected active and the current sequential time frame falls in the pattern's scaled duration, the pattern will be called active to produce its behavior. The use of an active pattern requires the copy and removal of the pattern's structures at its two duration boundaries. At the start, the previously recorded pattern structures are copied to the system's control environment, which produces the pattern behavior. At the end, the pattern structures are removed from the system. The removal of a pattern recovers the previous control environment before the pattern was copied.

The ordering mechanism controls the order and connection of the behavior patterns. It checks which pattern should be placed first in a sequential behavior and so on, and whether the ordered patterns are connected without blank interval in between two adjacent patterns. Adjacent connection between two patterns can be better tested after the scaling test for the previous pattern. In that case, we can better estimate the start time for the next pattern, especially when the duration of a pattern is determined upon a local environment reference. A set of patterns, in general, is arranged in a sequential time order, but the case for overlapping patterns whose behaviors are then combined in parallel is also possible. A list of behavior patterns can be repeated using a looping control structure, if required. One example of possible ordering of three behavior patterns is shown in Figure 5.7.

One example of using behavior patterns is the experience gained by performing a motion multiple times. This behavior difference can be modeled using behavior patterns. The first experience of avoiding obstacles in the path selects a pattern showing a timid and cautious behavior. The second and subsequent experiences selects a pattern showing a more confident behavior. The experience can be modeled by ordering the proper behavior patterns for the proper periods. The proper time to order the next pattern can be

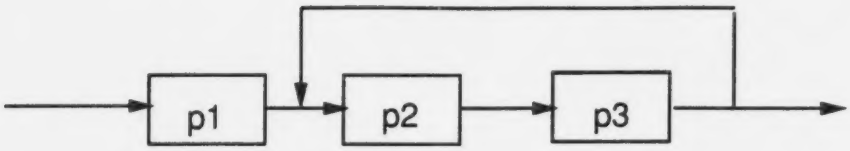


Figure 5.7 A Possible Ordering of Three Behavior Patterns

determined by the active duration selected for the current pattern. For instance, if the first pattern is called "timid" and the second pattern is called "confident", two sequential behaviors timid→confident→ and confident→timid→ can be ordered, with each adjustable behaving period.

5.2.3. Other Suggestions

One suggestion for using the relation hierarchy is to have a flexible control interface that allows the user to directly interact with the levels of the hierarchy. The user can visually experiment with different control levels, such as the selection of relations, the desirable partition of state sets, the grouping and scheduling of relations, and the sequential ordering of previously modeled behavior patterns. As one of our research goals, we are searching for a model that can be effectively and flexibly used for a wide range of behaviors. Part of the model is the relation composition hierarchy, which assists the user to freely express his or her ideas about how a motion behaves. This model extends the system's ability to a wide range of dynamic behaviors in the same or slightly varied environment.

Subjective control in motion, in terms of an object's mood, personality, and intention, is one important issue in today's animation research. In our proposed model, this issue is handled by the use of different control levels in the hierarchy. A character or a subjective control can be modeled through the hierarchical levels, such as a patient or a routine-like behavior. However, it could be promising to supply one additional

control level, subjective control level, on the top of the hierarchy. This level is aimed to supervise the subjective controls through the lower hierarchical levels, in an explicit and effective manner. A better control interface to view and revise the relevant subjective controls could be provided at this level. Currently, this level is not explicitly outlined in the relation composition hierarchy.

5.3. Relation Processing Algorithm and Its Complexity

This section presents a general algorithm for processing the relations at each time step of the animation. Here, we assume that the relations have been described using the ORS language (see Section 6.2) and structured through the interactive relation control system (see Section 6.3). The structured controls as proposed in the relation composition hierarchy (see Section 5.2.2) have been recorded in the system's data structures. These controls include the selection of a relation, a state control linked between two relations, a grouping and/or scheduling structure, and an ordered sequence of behavior patterns. As recorded in the system's internal data structures (see Figure 6.7 and Figure 6.8), the control information can be directly accessed while processing the following algorithm.

This algorithm runs whenever the user presses the recording button on the screen, while interactively structuring the relations using the interactive control system. This can occur at the time of selecting the relations, modeling the level structures, or completing an ordered sequence. The algorithm runs according to the control information recorded to the system and for each keyframe recording of an animation.

```
INPUT:
    relations & structured relation controls

OUTPUT:
    a sequential motion behavior

PROCESS:
    for each of the behavior patterns
     if the pattern is active
        if the pattern's starting frame f1 matches the current motion keyframe
            copy the pattern's structure to the system
        if the pattern's ending frame f2 matches the current motion keyframe
            remove the pattern's structure from the system
```

```
for each of the relations
 if the relation is selected
     if the relation is grouped to a time instant
         if the grouped instant matches the current motion keyframe
             change the relation's state to potential
     if the relation is in the potential or active state
         process the relation, possibly using group control
     if the relation is in the active state
         check for state control(s), if any
             issue the activating control(s), or
             issue the deactivating control(s), or
             issue the blocking control(s), or
             issue the terminating control(s)
         check for scheduled relations, if any
             change the scheduled relations' states to potential
             (at either the first active time, or an in-between time,
              or the last active time)
```

Figure 5.8 General Relation Processing Algorithm

In the above algorithm, the statement "process the relation" calls the relation routine in the library generated by the ORS language preprocessor. A block diagram of the relation control process is shown in Figure 5.9.

During the relation control process, two conditions can lead to the end of a response. One is a false enabling condition and the other is the end of the response duration. In other words, a response ends if its enabling condition becomes false or its response duration is over. The last code section in the relation process, also called the finalization section, is reserved for specifying the end control which is not applicable to a group of objects. This section only contains a small portion of the relation process and is separated from the rest of the process which may be combined with the group control facilities.

A relation is specified between two objects, the source object and the responder object, and each of these objects can be either an individual or a group. If a relation has a source group and a responder group, the relation control process is extended to the use of group control facilities, as shown in the following diagram:

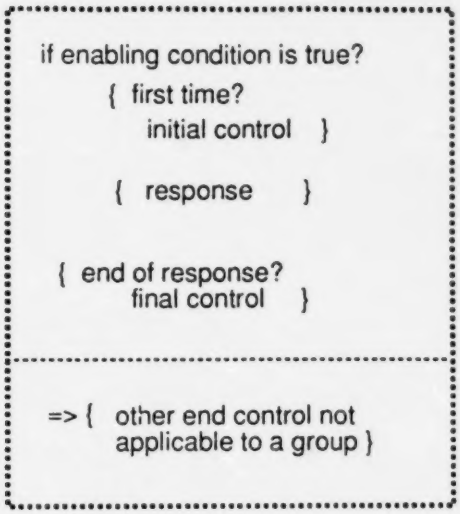
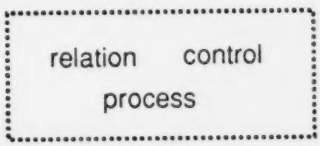

if enabling condition is true?
{ first time?
initial control }
{ response }
{ end of response?
final control }
=> { other end control not
applicable to a group }

Figure 5.9 Block Diagram of the Relation Control Process

for each member in the responder group
for each member in the source group



These group control facilities are automatically supplied by the system when a relation is called for processing.

If a relation is specified between a source group and a single responder, only the source group control is added to the basic relation control process, and if a relation is specified between a single source and a responder group, only the responder group control is added. No additional group control is required, if a relation is specified between a single source and a single responder.

The process for changing the state of another relation, using a state control or scheduling or grouping, is based on the data structure allowing direct access to that relation. The access time to another relation linked from the current processing relation should be a constant $O(1)$. Thus, the processing time for the step "for each of the relations" is only proportional to the number of relations used in the application.

The evaluation of relation processing algorithm is based on the assumption that the process is issued after a sequence modeling. Assume a scene animation application having R relations and using B behavior patterns in a selected sequential order. The lower bound of time complexity for each processing step of the animation occurs when all the relations are specified between two individual objects. In this case, each processing step takes $o(R+B)$ time. This estimation gives the best processing time. It could be worse when a relation is specified from or to a group of objects. The time required to determine the members of a group is a constant $O(1)$, based on the assumption that the member information has been stored in an object table which can be directly accessed from each processing relation.

The worst case for group control is hard to estimate because the size of the source group or the responder group of the relations can vary from one application to another, depending on the environment. If we assume that the same group members in the source group and the responder group are used for each of the relations, then the upper bound of time complexity will be $O(RM_sM_r+B)$, where M_s denotes the number of members of the source group and M_r the number of members of the responder group. This worst case will rarely occur because of the small possibility for the assumed uniform member distribution in all the relations.

A better way to estimate the time complexity is the average case. In this case, the complexity analysis is based upon the average number of group members. Under this assumption, if we assume M_{as} is the average number of members of the source group and M_{ar} is the average number of members of the responder group, then we have the average time complexity as a time function of $(RM_{as}M_{ar}+B)$. This average complexity at each processing step more closely represents the need of various applications. The use of minimum and maximum number of relations in a general environment context is given in Section 4.5.

Our algorithm checks for each relation for its state status at each processing step. This linear checking could be reduced to only the necessary relations whose states allow them to actively participate in the current motion control step. One such suggestion is to use three double linked lists for each of the potential, active, and suspended states. In this case, only the relations in the potential and active states need to be processed at each step. However, additional time for searching and updating these link structures as well as the space for maintaining the structures is required, which may not necessarily reduce the entire processing cost.

Chapter 6

Prototype Implementation

This chapter presents a prototype implementation of the relation control model. This system is not limited to one motion application, but can handle a wide range of applications. Our system uses the new control concepts and mechanisms proposed in the relation control model to model a wide range of behavior applications in dynamic environments. Both the modeling of relations and the relation synthesis hierarchy for modeling environmental behaviors are used in this system, which allow the user to explore previously unknown motion behaviors in unknown environments.

6.1. System Architecture

The initial motivation for designing the system is to provide the user with an intuitive, creative, and structured control environment for modeling behavior in a dynamic environment. The system is intuitive, providing the user with a natural logic for behavior control; creative, to allow the user to freely express his or her ideas, and structured, to guide the user interactively in modeling complex behavior through hierarchical structures. The system provides a general test-bed for environmental behaviors and supports the users' creativity at various control levels.

The main features of this implementation are:

[1] The system handles a wide range of unknown behaviors in unknown dynamic environments, in contrast to traditional systems which have predefined motions with certain behaviors in previously known environments.

[2] The system provides a mixture of language description and interactive control. The language description allows for the general and natural control of an animation, and interactive control allows for the flexible modeling and editing of an environment and environmental behavior.

[3] The system provides multiple behavior understanding levels, varying from a simple level to a more

complex, responsive, and individual level in dynamic environments. These levels enable users with different animation knowledge and skills to learn from the experience of using the system at a suitable level of detail.

[4] The system presents the user with a structured control interface, not detailed control codes, to visually trace and experiment with an object's behavior. With a simplified interface, the user can quickly model a motion and revise the motion for other intended behaviors.

Our implementation is called the relation control system (RCS). The system supports a work place combining language description and interactive control. The language part of the system is used to model relations, in terms of the enabling condition, local control properties, and response behavior for each relation. The interactive part of the system assists the user with structuring the model and environmental behaviors based on previously described relations.

The two parts of the system, language description and interactive control, are ordered in a processing pipeline: language processing followed by interactive control. The first processing step of RCS interprets the language description of objects and relations for the next step of the system, which allows the user to interactively compose an environment and structure the relations from simple primitive behaviors to more complex environmental behaviors. Both steps contribute to the modeling of complex behaviors. Here, the language processing is not used for describing a complete motion, and the interactive control is not used for issuing predefined motion sequences or adjusting a few parameters in a predefined motion. In an interactive session there are many ways of combining the relations. These alternatives can be explored by the user in the second interactive processing step, using the relations described by the language.

The input to the first processing step is a language description of objects and relations, and the output from this step is a set of system modules built from the input data. The input to the second processing step is the modules from the first step and the system's interactive facilities, and the output from it is the modeled environment and environmental behaviors, which are displayed interactively through recorded frames. The inputs and outputs of the two steps in the RCS system are outlined in Figure 6.1.

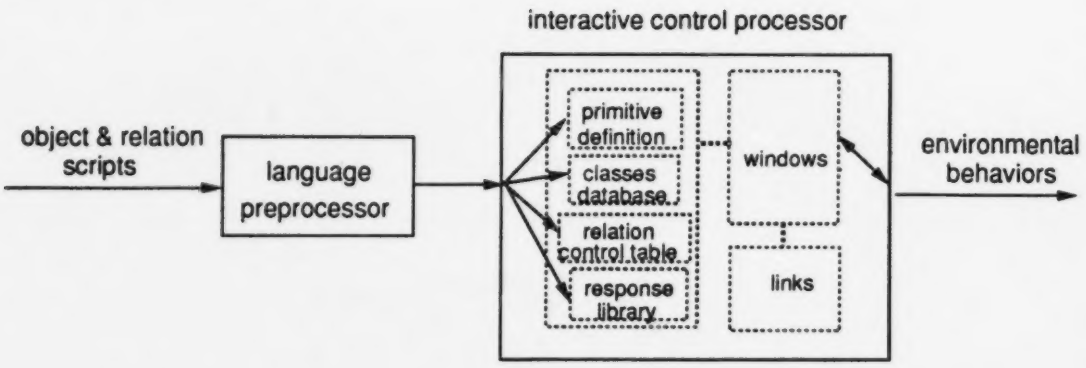


Figure 6.1 Input and Output Process Interface for RCS

Modules generated by the first processing step include a primitive definition file, classes database, relation table, and response library. The primitive definition file contains the user defined object types and instances declared from the standard and defined data types. The classes database stores the objects modeled by the class types, including polygons, curves and surfaces, fractals, and particles. Procedurally modeled objects are included in the class type of polygons, where the object names recorded in the polygon database are actually the names of the routines for generating the objects. The relation table contains each relation's interactive control properties, including the relation's name, source and responder object names, its enabling condition, initial sensing state, priority assignment, local responsive control parameters, and optional switches. These control properties are the ones specified in the control header of the relation frame. The response library contains the action part of each relation, which can be called by the relation's name recorded in the relation table. The action part of a relation includes the procedural description in both the action body and finalization sections of the relation frame.

The interactive part of the system can be divided into four components. These are the scene composing component, pattern control component, sequence control component, and scene rendering component.

These components cover the three processing steps of computer animation: modeling, motion control, and rendering. The motion control part is covered by both the pattern control component and sequence control component. The division of these two components resembles the level division of the relation composition hierarchy, where the pattern component covers up to the pattern control level, and sequence component covers only the sequential control level of the hierarchy. The reason for using one component for several control levels is to address the lower control levels in a consistent control interface and provide names for these basic behaviors. The named behaviors are then used as the basis for the sequences constructed in the sequence control component.

The main functions of the four components are:

Scene composing component

The composing component is used to interactively compose a scene environment based on the objects previously described in our relation language. This composition is mainly done through mouse interactions, such as menu selecting, object picking, object dragging, and button clicking.

Pattern control component

The pattern control component is used to interactively structure relations, previously described in our relation language. The structuring uses the mechanisms proposed for the selective, state, and pattern control levels. This component also traces and displays the structures built among relations dynamically during a motion's progress, in a simplified graphical display. The experience of visualizing the dynamic development of control structures helps the user to learn from previous experiments.

Sequence control component

The sequence control component is used to interactively structure sequential behaviors based on the patterns modeled in the pattern component. The structures that are modeled in this component are pattern selection, pattern ordering, time scaling, and sequence recording and playback.

Scene rendering component

The rendering component is used to adjust the scene views and special coloring effects.

The interactive environment of RCS, including the modules generated from the language processing step, the four interactive components, and the standard system facilities, is outlined in Figure 6.2.

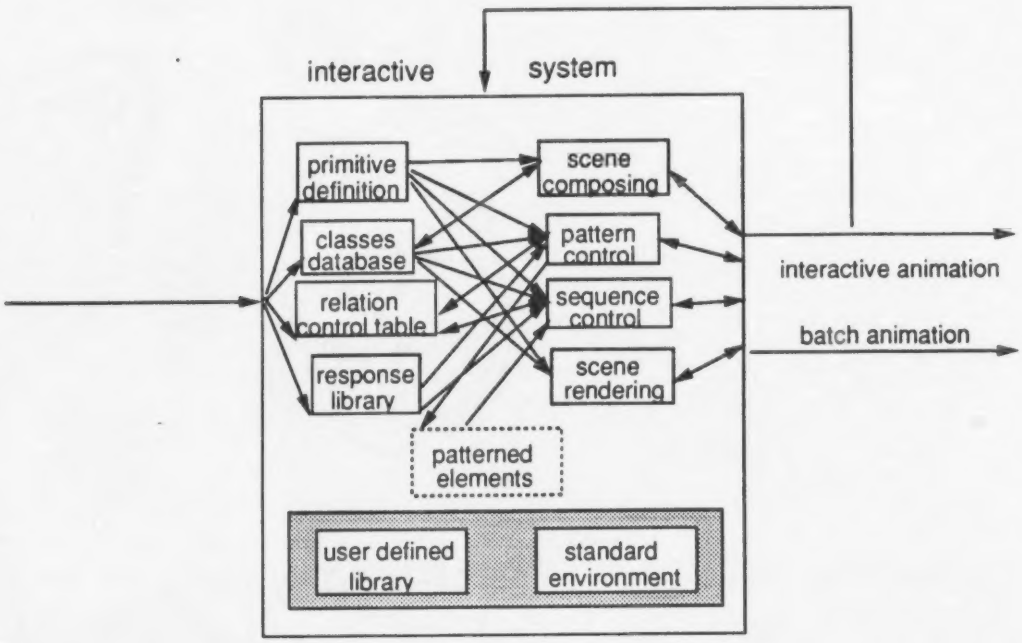


Figure 6.2 Structure of the RCS System

The following sections describe the language used to describe objects and relations, and the interactive components of the environment used for modeling behaviors. The discussion of each system component emphasizes the control interface between the system and user, showing how the user interactively composes a scene environment and structures a sequential behavior according to the environment, and how the controls specified through the interface are directed to the system's internal modules.

6.2. Objects and Relations Scripting Language

The objects and relations scripting language (ORS language) is designed for describing both the objects used for composing the environment and the relations among the objects that are used in a behavioral animation. The language combines C procedural descriptions and frame-like descriptions for modeling the objects and relations. The procedural statements in C are used as the basic elements to fill in the modeling details in a frame-like format. The modeling of objects is used for procedurally defined objects. The modeling of relations is based on the relation frame proposed in Chapter 5. This frame consists of three sections: control header, action body, and finalization. The control header section contains a set of slots, each of which can be specified by either a value, a variable name, or an expression in C. The action body and finalization sections include mainly the C statements and other specially defined relation expressions.

The major difference between the ORS language and other animation languages is the relation description. The frame description of a relation only relies on two objects, the source and responder. Thus, a relation is described in a local, simple, and independent way, while in other languages a motion is usually described in a global, complex, and dependent way. The description of a motion through programming can be very difficult, time consuming, and inflexible. The description of a relation only involves the control details for the two related objects. It is also independent of the way the relation is used in a behavior application. The simplified description of a relation does not limit its power to model complex motions. On the other hand, the use of simply described relations gives us more freedom to create a motion with variable behaviors.

The modeling part of the ORS language covers both procedural and other standard way of modeling objects. The modeling of standard objects, such as polygons, curves and surfaces, fractals, and particles, is described by the common features of each standard type. For instance, the model of a polygonal object includes a list of vertices, assigned to the slot "VERTICES: x1 y1 z1, ...". The model of a curve or surface object is based on a collection of control vertices, curve or surface type, and other control parameters. A

fractal object is modeled by the vertices of a triangle or rectangle primitive, and the other control parameters required by a stochastic process. A particle object is modeled by a set of particles or vertices, plus the initial position, velocity, life time, and other properties of the particles. The routines for generating the objects of these standard types are automatically supplied by the system. The syntax for the standard object types is presented in Figure 6.3, in which upper-case words are ORS reserved words and lower-case words are values supplied by the programmer.

```
OBJECT NAME: a_name   /*  polygon object
TYPE: a_number       /*  POLYGON | POLYLINE
COLOR: a_number      /*  RED | 4
VERTICES: x y z , . . . ,
      . . . ;
COLOR: a_number
VERTICES: x y z , . . . ,
      . . . , . . . ;
  . . .

OBJECT NAME: a_name   /*  curve & surface object
TYPE: a_number       /*  CURVE | SURFACE
C&S TYPE: a_number   /*  BEZIER
STEP: a_number
COLOR: a_number
VERTICES: x y z , . . . , /* 4 vertices for a curve
      . . . , . . . ; /* 16 vertices for a surface
  . . .

OBJECT NAME: a_name   /*  fractal object
TYPE:  a_number      /*  FRACTAL
VERTICES: x y z , . . . , /*  either a triangle or
      . . . ;        /*  a rectangle
COLOR: a_number
RATIO: a_number
LEVEL: a_number
  . . .

OBJECT NAME: a_name   /*  particle object
TYPE:  a_number      /*  PARTICLE
VERTICES: x y z , . . . , /*  ie. 4 vertices
      . . . , . . . ;
VELOCITY: v dx dy dz , . . . . , /* speed & direction
      . . . . , . . . . ;
COLOR: c1 c2 c3 c4
SIZE: s1 s2 s3 s4
LIFETIME: a_number
```

. . .

Figure 6.3 Syntax for Standard Object Types

Procedural modeling is generally used for user-defined objects. The modeling syntax used for this consists of a set of slots, which are: name, primitive, para_def, and initial. The slot "name:" accepts an object's name. The slot "primitive:" contains a description of a user defined object type and an instance declaration. The slot "para_def:" collects a set of initial values of parameters used for modeling the object. The slot "initial:" specifies the routine for assigning the initial parameter values to the declared object instance. The syntax of the procedural object type is given in Figure 6.4.

```
OBJECT NAME: a_name
PRIMITIVE: {
        /* primitive data structure
        /* of the named object
        . . .
      }
PARA_DEF: double a_name a_real_number ,
        . . .
      int a_name a_int_number
        . . .
INITIAL: {
        /* initial assignment of the
        /* parameter values to the modeled
        /* object type
        . . .
       }
```

Figure 6.4 Syntax for the Procedural Object Type

Modeling of an individual object and a group of objects is treated in a similar way in the ORS language. The only difference between the two cases is the inclusion of a pair of brackets. If a pair of brackets is included after the object name (such as birds[]), the object is a group; otherwise, it is assumed to be an individual. Note that no actual group members are given in the case of a group declaration, which is left as one control option in the interactive control process. The actual group members are determined by the com-

position of the environment. Each time an environment is changed, the actual group members currently used can change.

The procedural description of initial modeling, in the slot "initial:", is independent of the object types. No matter whether the object is declared as an individual or a group, only the object name is referenced in the description. If the object is declared as a group, the additional control structure to apply the modeling code to each of the group members is automatically supplied by the system, when the routine is called. Since the actual group size is not known at the time when the code is specified, a variable whose value is the size of the group is given as a parameter. A procedural modeling example of a group of blocks is given below.

```
object name: block
primitive: {
        struct blockobj {
            double x, z;
            double hw, hl, h;
            int color;  };
        struct blockobj block[]; }
para_def: double sx 0.1, sz 0.1, sh 0.3,
            sw 0.2, sl 0.2,
           int   sc 28;
initial: {
        block.x = sx;
        block.z = sz;
        block.hw = sw;
        block.hl = sl;
        block.h = sh;
        block.color = sc;
      }
```

The motion part of the ORS language uses the relation syntax, defined in Chapter 5, to describe the motion produced by the relation. The slot values in the control header of a relation frame can be constant values, variables, or expressions in C. The descriptions in the other two frame sections, the action body and finalization, consists of procedural statements plus other relation control statements and references to slot values specified in the control header. The description block for each of the two sections is formed by a pair of braces {}, consistent with the convention of the host language C. Each block can be identified by the frame part context, where the action body section follows the relation's name, and the finalization section

follows the "==>" symbol after the action body.

A slot description in the action header section of a relation could be missing. In this case, two rules are used depending on whether the relation is the first one in the motion part. A missing slot description in the first relation is filled with a default slot value by the system. However, this substitution rule is not always true for all the slots. Some slots may not have a default slot value, such as the source object name or the responder object name. In this case, the slot should be explicitly assigned a value on its first use. The second rule is to inherit the slot value from the previous relation, if the current relation is not the first one declared in the motion part. Otherwise, an error message will be produced.

The slot values in the control header can be referenced either by slot name or parameter name in the action body and finalization sections. For instance, the enabling condition specified in a sensing channel slot can be referenced by the channel's name in the control code. The excitatory response for a channel is expressed as the channel's name, for example: "channel name: { ... }", and an inhibitory response is expressed as "!(channel name): { ... }". Collective control based on multiple channels can be expressed using the connectors AND or OR to form an extended channel reference. All the channel names used in the procedural code are replaced by the sensing conditions from the control header, when they are processed by the system.

The finalization section is mainly used for summarizing a group's response and issuing the necessary control at the end of a response. The group summary control may direct a specially selected group member or produce a motion based on the average group response. Other controls in the finalization section include state control calls either to the relation itself or to other relations, message passing, or an environment update.

Objects modeled by one of the standard types (polygons, curves and surfaces, fractals, and particles) are rendered by the standard system routines. Only the objects whose types are defined in the modeling part need to include a rendering description. The same rule for handling the difference between an individual and a group in the modeling and motion control parts is used for the rendering part. That is, an object

declared as a group is treated the same as an individual. As a result, the rendering code of an object is specified only in reference to the basic object type. Additional group control in rendering is automatically produced by the system. For instance, the rendering code for a group of blocks can be as simple as:

```
block {
      drawblock(block_i,Scene);
     }
```

where block_i is the group indexing variable, Scene is the built-in data structure for display, and drawblock() is the rendering routine for the block object, in the user-defined library.

6.3. Interactive Control System

The second step in the pipeline is to structure relations into a desired behavior using an interactive control system. This system is constructed in such a way as to support an intuitive, creative, and structured control of behavior animation, based on the relations modeled in the first step. To this end, four interactive components are supported by the system, which are: scene composing, pattern control, sequence control, and scene rendering.

Interactions in a component are supported by the use of windows, such as menu window, message window, scene window, button window, and text dialogue window. By interacting with these windows, the user can select a menu item, adjust a parameter value, pick up an object and drag it to a new scene location, and press a command button. Menus for each component are organized in a hierarchy. One example of a menu hierarchy used in the scene composing component is "scene composing ==> class/object/value". In traversing a menu hierarchy the following rules are used: a submenu is opened if the corresponding item on the parent menu is selected; and a submenu is closed if the "exit" menu item is selected. More details on the contents of the menu hierarchy are given in the sections on each component. Figure 6.5 shows a global view of the whole menu hierarchy for the four interactive components of the system environment.

The interactive system environment was built on a IRIS 3130 workstation. There are four interaction areas: display area, menu area, system message area, and on-line help area. In this basic division, the

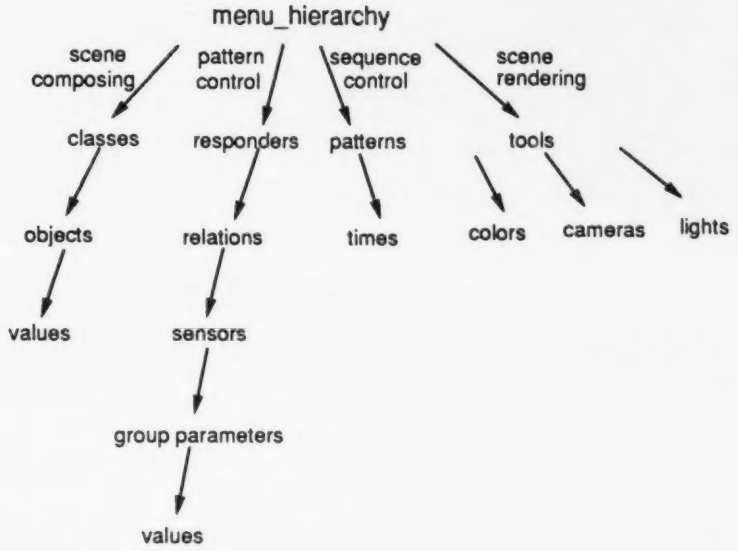


Figure 6.5 Menus Hierarchy For the Four Interactive Components

display area shows a three-dimensional view of the composed scene environment and the motions in the scene; the menu area displays the active menu; the system message area reports the current system control state and any error messages; and the on-line help area provides guidance for the user. The division of the screen into four interaction areas is shown in Figure 6.6.

Most interactions are conducted using the mouse on the IRIS workstation. The three buttons on the mouse are referred to as the left_button, middle_button, and right_button. A combination of the buttons is expressed as a combination of the buttons names, such as left_middle_button. The use of mouse buttons depends on the context of the interaction. For example, in menu interactions, pressing the left_button on a menu item traces down or up one level of the menu hierarchy. Pressing the middle_button on a menu item selects the item. Pressing the right_button on a menu item executes the control defined by the item, such as the control of a relation.

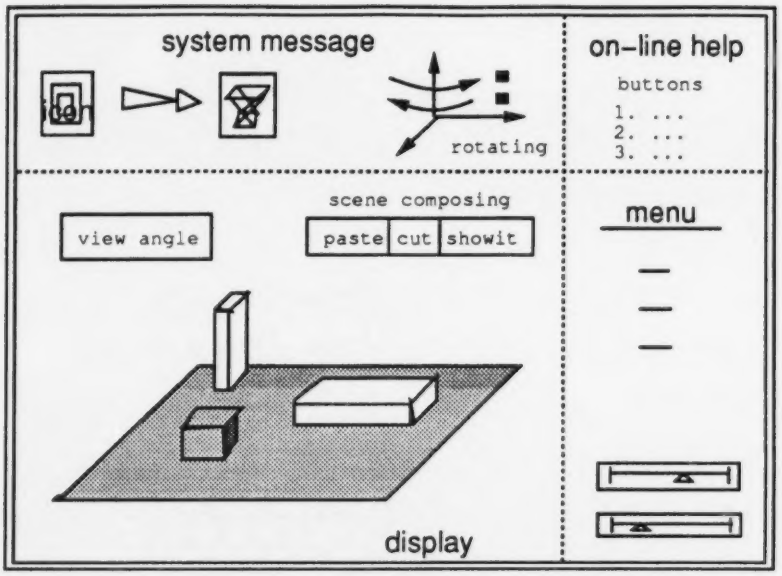


Figure 6.6 Screen Division For the Four Interaction Areas

The interactive control environment is built using the WINDLIB package and FDB package [Green86a, Green86b], which supports device-independent window hierarchies, frame-like graphical databases, and three-dimensional display structures. The high-level window and content structures of the WINDLIB package are directly linked to the low level graphical routines of the IRIS workstation, to speed up the display process.

6.3.1. Scene Composition Component

The scene composition component is used to interactively compose a scene environment when a set of dynamic behaviors can be tried out. Composing an environment is the first activity in modeling behaviors. The selection of this component, by a mouse click in the component icon, brings up the four interaction areas for this component. The display area shows an empty scene environment the first time the component is used. Otherwise, an environment previously composed is displayed in the area. The menu area shows the class menu as the root of the menu hierarchy for this component. The system message area shows the menu levels that have been traversed so far and other system messages.

Mouse functions in the menu area are: clicking the left_button on an item traces the menu hierarchy down one level from the item, clicking the middle_button on an item selects the item, and clicking the right_button on an item displays the contents of the item, such as an object if the menu hierarchy has been traversed down to the object menu. These general rules for using the three mouse buttons in the menu area may not apply all the time. If a case does not currently apply, it is skipped and nothing will happen upon clicking the mouse button. Clicking the middle_button on a menu item a second time reverses the previous action. It is like an undo function. If a menu item is previously activated, clicking the middle_button on the item a second time deactivates the selection. Clicking the third time will activate the item again, and so on.

A set of soft buttons are specially configured for modeling a scene environment. Three soft buttons are located at the upper-right corner of display area. These are "paste", "cut", and "showit". The use of these buttons simulates cut and paste interactions in a three-dimensional environment. In the upper-left corner of the same area, a soft button labeled "view angle" is used to vary the viewing angle of the scene camera model. A click on this button gives a new scene view rotated 90 degrees clockwise. In addition, two soft buttons placed in the system message area are used for either clockwise or counterclockwise rotation of a selected object.

The use of these soft buttons and other mouse interactions allows the user to interactively select an object from the object menu, resize the object by adjusting its modeling parameters, place and replace the

object in the test scene, set a new initial object orientation, or add the object to or delete the object from the final environment. More details on these interactions are given in the scene component section of Appendix A1.

The final environment can be viewed in any of four directions, front, left, back, and right, respectively. The front view is the default one on initially entering the environment. This view can be changed to other views by mouse clicks on the "view angle" button. Each click on the button changes to the next view.

Composing an environment in our system can occur at any point in the animation process. Since we use relations for modeling a scene motion, the task of composing the complete environment can be delayed until after the motion control process. Here, we propose a two-step composing strategy. The first step composes the basic environment with the objects of different types. This basic environment is used for modeling the behavior elements in response to the basic object types. After that, the basic environment can be extended by duplicating existing objects. Note that the environment produced in the second step does not require new behavior elements different from the ones modeled in the basic environment. It only requires the use of the basic elements.

6.3.2. Pattern Control Component

The four interaction areas of the pattern control component are initially as follows: The system message area shows the levels of the menu hierarchy that have been traversed and other system messages. The on-line help area prompts for user input and outlines the mouse button functions in the menu area. The display area shows the composed environment as well as the motions in the environment. The menu area initially lists the "responder" menu in the component's menu hierarchy, which can be traversed up or down through mouse interactions.

Three control modes are used in the menu area of the pattern control component. These are the menu mode, group mode, and schedule mode. The menu mode is the one normally used in the menu area of other components. In this mode the following operations can be performed: A left_button click on a menu item

traverses down or up one level in the menu hierarchy. A middle_button click selects the menu item. A right_button click displays the item's content depending on the menu context. Both group and schedule modes are used for interactively structuring relations from the relation menu. The detailed use of these two modes is explained in the section on structuring relations. A quick overview of the use of the three buttons in each mode can be obtained from the submenu for each mode, shown in the on-line help area. The submenu of each mode is opened if a left_button click is performed on the mode item and the item is active.

The "responder" menu is first displayed in the menu area when the pattern component is selected. In this menu, only the moving objects modeled by the ORS language are listed, with the ones in the environment highlighted. Traversing down from a responder item, relations using the object as a responder are listed in the menu "relation". Among the relations listed, the relations selected by the environment are highlighted. These relations are automatically selected by the system when the pattern component is opened, according to the source and responder objects present in the environment. The relations automatically selected by the environment can be interactively selected again by the user. The user selection is made by clicking the middle mouse button. This interaction allows the user to undo the previous selection, giving the user an opportunity to reselect the relations.

The menu hierarchy of the pattern control component is: "responder/relation/sensor/grouppara/value". Down from the "relation" menu, the "sensor" submenu presents the four standard sensing channels (visual, smell, sound, and tactile). The ones used by the parent relation are highlighted in the menu. Further down, the "grouppara" menu lists the group parameters specified under each sensing channel. The "value" submenu displays the values initially assigned to the parameters of the parent group. These values can be interactively adjusted by the user. The parameter grouping facility collects the parameters into a two-level naming hierarchy, which allows quick viewing and editing of parameters. By traversing the menu hierarchy, the user gets a high-level view of how the relations are controlled by each other's local properties. This view helps the user to adjust a relation's local behavior and structure the global behavior of relations.

In the current system, state control is mainly modeled using the state control statements in the relation frame description. The transitions between the active state and other states such as potential, blocked, and terminated, are described in the action code and finalization parts of a relation frame. These statements implicitly structure the state control between two relations, from the one issuing the call to the one being called. These calling structures are dynamically displayed in each control step using a graphical illustration. In this notation, a relation is drawn as a double circle, where the outside circle represents the relation's sensing and the inside circle the relation's response. Initially, when a motion starts, all the potential relations are drawn in green in both circles. If a relation is in its active sensing mode, its outside circle turns red. Similarly, if a relation is in its active response, its inside circle turns to red. State control is shown by a line pointing from the relation issuing the call to the relation being called. The type of the call is shown by the use of line drawing styles and colors.

Structures in the pattern control level can be interactively produced using the control modes "group" and "schedule". Detailed interactions for structuring relations under these two modes, as well as for editing a pattern structure and recording and viewing a structured pattern, are given in the pattern component section of Appendix A1.

The structures among the relations produced by this component are recorded in the relation table. Entries of each relation recorded in the table include the relation's selection by both the environment and the user, the relation's state that could be changed by its enabling condition and structured state controls, a grouping or a scheduling, and its variable local control properties. The relation table is implemented as a linked list, for each responder object. The internal structure of the relation table is shown in Figure 6.7.

At each control step, relations associated with each moving object (responder object) in the environment are searched. If a relation is currently in the potential or active state, the relation performs a sensing or respond action. A relation automatically performs its response if its enabling condition becomes true, and updates the relation's state to active. When either the source or responder is a group, the group control structures are automatically checked and constructed when the relation is called.

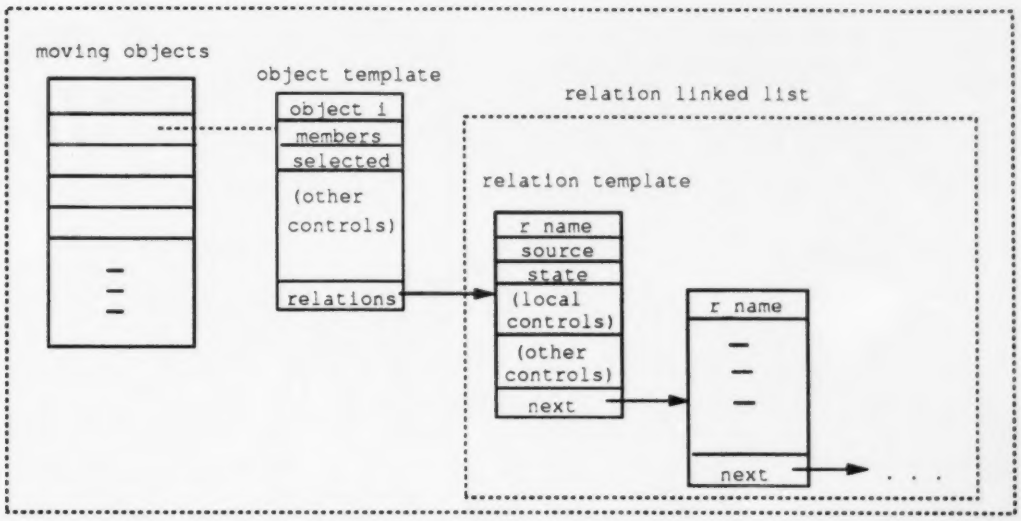


Figure 6.7 Internal Data Structure of the Relation Table

A grouping structure is checked when the time for the group is reached. If this time matches the start time, the relation's state is changed to active and the relation is called by the system. A scheduling structure is checked by the name list linked to the referenced relation record. If there is one, the scheduled relation's state is changed to active and the relation is called with its local control parameters.

Once a behavior pattern is on-line tested to the user's satisfaction, the pattern's structure can be recorded to the system for the later use. The interactions for recording a pattern are as follows: A dialogue box appears after a right_button click on the title of the "responder" menu. The user is asked for the pattern name and confirmation of the recording. At this point, a cancellation is still possible if a negative answer is given. A recording following a positive answer copies the relation structures to the pattern's name. In either case, an answer returns the system back to the original state, from which a new behavior pattern can be modeled.

A pattern is recorded with only the relations selected by both the environment and the user when the pattern is modeled. Each of the relations is recorded with its local control properties and other structural information modeled with the relation. These include the relation's state, and grouping or scheduling structures. The relations selected for a pattern, along with the necessary structural information are stored in a linked list, as shown in Figure 6.8.

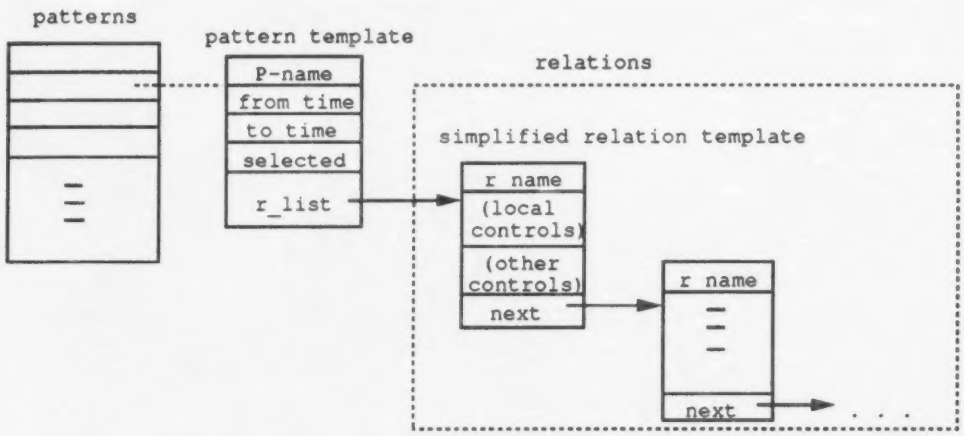


Figure 6.8 Internal Data Structure of Patterns

6.3.3. Sequential Control Component

The sequential control component is used for modeling new sequential behaviors using the set of behavior patterns previously modeled in the other system components. This component implements the sequential control level of the behavior hierarchy, where the only elements used for structuring a sequence are behavior patterns. These patterns can be ordered and scaled along a sequential time dimension, and recorded to the system for a quick motion review. Initially, there are four interaction areas in this component. The current path through the component's menu hierarchy and other system messages are shown in the system message area. Legal mouse functions for the menu area are displayed in the on-line help area.

Motions can be recorded and viewed in the display area. The first menu in the menu hierarchy, "sequential control => pattern/time", is listed in the menu area.

Three soft buttons in the display area are used to extend the current environment to a more general environment. These buttons are "pick", "move", and "cancel". The process of composing a general environment is divided into two control modes: picking and moving. Detailed interactions for using these modes, as well as for structuring sequences from patterns and a sequence recording and viewing, are given in the sequence component section of Appendix A1.

The current system mainly considers linear structures for sequential behavior. If a parallel or partial parallel structure is used for modeling a sequence, a different structure copy and remove strategy should be used. This strategy should consider the overlapping case when two or more patterns are used in parallel, rather than one at a time. The copy of an active pattern may or may not overlap part of the structures previously copied. It may cause confusion about the possible overlapping part, when the previous pattern is removed. One solution to this problem is to let the system keep track of the overlapping parts and use this information to prevent errors when recovering an active pattern.

6.3.4. Scene Rendering Component

The scene rendering component is used to adjust the rendering effects of the scene environment, including color mapping, camera models, and lighting models. These rendering controls are divided into separate rendering tools. The use of a tool is signaled by a left_button click in the tool icon, which opens the interactive control environment for the tool. If the color mapping icon is selected, four interaction areas are assigned as follows: the system message area shows the levels of the colors menu and other system messages. The on-line help area describes the legal mouse interactions in the menu area. The menu area lists the commands used for color mapping. The display area presents a test-bed for selecting colors, which includes three standard color sliders, a sample color panel, and a set of color panels for recording the composed colors. A color panel is a small rectangular area for showing an assigned color.

Commands listed in the "colors" menu are *copy, uncopy, forward, backward, nextpage, and backpage*. The *copy* command copies the current color to a color panel. The target of the copy is indicated by a pointer that can be moved back and forth among the color panels. The *uncopy* command reverses the last copy action. The *forward* command moves the pointer from the current panel to the next panel, and *backward* command moves the pointer from the current panel to the previous panel. The *nextpage* command opens a new page of color panels, and the *backpage* command retrieves the previous page of color panels.

In the display area, a color is composed using three standard color sliders, ranging from 0 to 255 in red, green, and blue. The value of a standard color can be selected by a mouse click in the color slider. Every time a new value of a standard color is selected, the newly composed color is formed and displayed in the sample color panel. Our current system uses the standard RGB system for its color composition. This color standard could be converted to other color standard systems, such as CMY (cyan, magenta, and yellow) or HSV (hue, saturation, and value).

Initially, five standard colors (black, white, red, green, blue) are displayed in the first row of color panels. These colors can't be changed. The mapping of a selected sample color to a color panel follows the steps of selecting a panel and copying the sample color to the panel. For detailed steps refer to the rendering component section of Appendix A1.

Four camera settings, camera1, camera2, camera3, and camera4, are used for setting up the multiple views of the environment and motion on the screen. These can be the long shots from different viewing angles, closeup shots for selected details, or parallel shots of related control effects. A camera setting is defined by a set of viewing control parameters, including the eye position, viewing direction, position aimed along the line of sight, and view volume in the eye coordinate system. By default, camera1 is used for a long shot view of the environment and motion, projected to the whole screen of the display area. This view is also used in the scene composing, pattern control, and sequence control components of the system. Besides the use of this camera, the other three cameras are used for alternative views of the same environment and motion in the viewing component. The viewing control parameters of all four cameras can be

interactively set by the user.

Some suggestions for the effective use of alternative cameras are as follows. Camera2 can be used as a closeup shot of a partial environment, which shows the main character's subtle reactions. Camera3 and camera4 can be used for two parallel events or a pair of related source and responder objects, to simulate the view of each object. The use of parallel views also illustrates the relation control ideas introduced in the environmental behavior animation. On the other hand, a camera as a special object can be animated in the viewing space, by either rotating or translating the camera model. Also, linear interpolation can be used to move the camera along a path.

The use of multiple cameras on the screen can be one of the four cases: one, two, three, or four cameras. If the one-camera viewing mode is selected, the view set by the selected camera is mapped to the whole display area. If the two-camera mode is selected, the two selected camera shots are shown on the screen, with each in half of the display area. If the three-camera mode is selected, the first camera view is displayed in the left-half area and the other two are each in half of the right-half area. If the four-camera mode is selected, each camera view is shown in a quarter of the display area. These subdivided viewports for each camera mode can be resized to a smaller area, using the mouse interactions.

The use of a grouped camera mode depends on the number of cameras selected for the current view. A camera can be selected by a middle_button click on the camera item in the camera menu. Initially, camera1 is selected by default, which implies the use of one-camera mode. To change this default mode to another mode, the user can interactively select the number of cameras required by the mode. Also, a quick way to adjust the viewing control parameters of selected camera group is possible, through the interface supported in the parameter menu. Namely, a left_button click on the "continue" command advances the current camera's parameter menu to the next camera's one, in the selected camera group. A left_button click on the "exit" command ends the cycle of quick group adjusting.

Interactions for viewing a previously recorded motion sequence in a selected camera mode are given in the rendering component section of Appendix A1. Currently, only the default camera mode is

implemented in the current system version.

The lighting model in the current system uses the one available from the IRIS graphics library, Gouraud shading of diffuse light reflection. Other shading effects, such as Gouraud shading of specular light reflection and Phong shading of diffuse light reflection, are not included in the current version, but could be added to the system.

Chapter 7

Examples of Dance Motion in Various Environments

This chapter presents a set of examples using the relation control model. These examples use ballroom dances as the basic animating behaviors. Based on these, other simple movements, such as forward step, backward step, side step, head turn (to the left or right), body turn (to the left or right), arm up and down, pause, changing to a new dance, changing to a new pattern, and following a pattern, are used as the alternative behaviors during a dancing. These basic and alternative behaviors are modeled by relations and used dynamically in an animation, upon each relation's sensing ability and the state of the environment. The examples are developed initially from a simple static environment, with obstacles and boundaries, then incrementally to other more complex and dynamic environments. Similarly, the behaviors in the environments can be incrementally modeled using the relations.

7.1. Dancer and Dance Notations

The main character used in the dance examples is an articulated human figure. It consists of sixteen linked segments: the head, neck, upper_body, lower_body, left_upper_arm, left_lower_arm, left_hand, right_upper_arm, right_lower_arm, right_hand, left_upper_leg, left_lower_leg, left_foot, right_upper_leg, right_lower_leg, and right_foot. The topology of these segments is illustrated in Figure 7.1. The arcs point from child segments to their parent segments in a tree-like fashion.

Every body segment, except the upper_body segment (which is defined as the root of the tree), has a parent segment. Every segment except the feet, hands, and head has one or more child segments. For instance, the lower_body segment is linked to its parent segment, upper_body, and its two child segments, left_upper_leg and right_upper_leg. In the dance examples, body segments are drawn as a three-dimensional model with body proportions measured from a live human figure. The front and back surfaces of each body segment are rendered in a different color intensity.

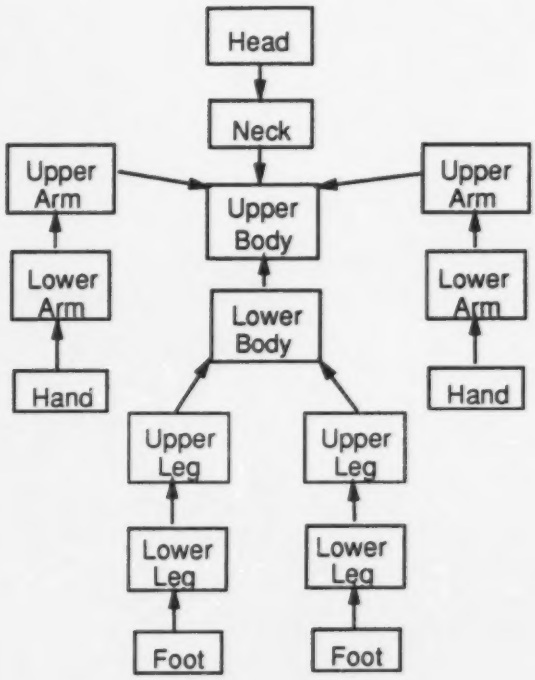


Figure 7.1 Tree-like Structure of Human Figure's Model

Two types of coordinate systems are used to control the figure's motion: the world coordinate system and local coordinate systems. The world coordinate system (also called the inertial coordinate system) is centered at the root node of the figure's tree structure. This coordinate system places the figure in the world space, which measures movement applied to the entire figure model. The local coordinate systems are centered about the point which joins each segment to its parent, which measures the movement applied to the segment relative to the coordinate system of the parent. Both the world and local coordinate systems have three orthogonal axes, x, y, and z, in a right-handed configuration. The length of each body segment is aligned to the y axis of the respective local coordinate system, except for the ones used for the foot segments which are aligned to the z axis.

The origin of the world (inertial) coordinate system is set to the joint between the upper_body segment and the neck segment, while the origin of each local coordinate system is set to the proximal hinge of the segment (the point where the segment connects to its parent). When a limb segment rotates around one or more axes, the three orthogonal vectors of the local frame attached to the limb also rotate to obtain a new orientation of the coordinate system. Thus, with the moving reference frame, each segment maintains a consistent description in its local coordinate system. The links between the body segments are connected as revolute joints with three degrees of freedom. The joint at the root of the tree consists of both three revolute and three translational degrees of freedom, and connects the whole tree to the world frame.

Each segment is defined by four parameters in its local coordinate system: the length of the segment and the three Euler angles with respect to its parent. The length of the segment is defined according to the figure model. The Euler angles of a segment with respect to its parent segment can be used to calculate a rotation matrix, composed of three orthogonal rotation matrices for the x, y, and z axes. For instance, the matrices for computing the Euler angles of a segment r with respect to its parent segment $r-1$ are

$$R_x(\Phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\Phi) & sin(\Phi) \\ 0 & -sin(\Phi) & cos(\Phi) \end{bmatrix}$$

$$R_y(\Theta) = \begin{bmatrix} cos(\Theta) & 0 & -sin(\Theta) \\ 0 & 1 & 0 \\ sin(\Theta) & 0 & cos(\Theta) \end{bmatrix}$$

$$R_z(\Psi) = \begin{bmatrix} cos(\Psi) & sin(\Psi) & 0 \\ -sin(\Psi) & cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R^r = R_z(\Psi) R_y(\Theta) R_x(\Phi)$$

where $\Phi, \Theta,$ and Ψ are the Euler angles for the rotations about the x, y, and z axes, which are in the order of z, y, x. The matrices shown above assume column vectors for calculating the rotations.

Similarly, the orientation of segment r with respect to the inertial frame can be obtained from the matrix product:

$$R_I^r = R^1 R^2 \cdots R^{r-1} R^r$$

The endpoint of segment r, the point which is not linked to the parent segment $r-1$, is expressed as

P^r in the frame of segment r. P^r can be calculated in the frame of segment $r-1$ by:

$$P^{r-1} = R^r P^r + O_r^{r-1}$$

where O_r^{r-1} is the location of the proximal hinge of segment r expressed in the frame of segment $r-1$, and P^{r-1} is the coordinates in frame $r-1$. Repeating the calculation can convert the expression P^r in the frame of segment r to a world coordinate P^0 in the inertial frame.

With the tree structure of an articulated figure, the standard dance steps such as forward step and backward step can be constructed using the positions and orientations of a figure's body segments. Once these standard dance steps are defined, a wide range of dances can be composed in the following way: a dance is composed of a set of patterns, a pattern is composed of a set of steps, and a step is composed of a set of well-defined body segment positions, such as foot positions, arm positions, head position, footwork, and body turn. Time beats required to reach these positions at each dance step may be considered as another control factor.

In standard dance notation, foot positions defining the dance movement of either the left (or right) leg are 1st, 2nd, 3rd, 4th forward, 4th backward, and 5th positions. Other additional foot positions are SIP (step in place), XIF (cross front), XIB (cross behind), and UNX (uncross). The arm positions in dance movement are either a close position (CP) with the dance partner or an initial standing position. The head position can be one of LL (look left), NP (normal position), and LR (look right). Footwork describes the subtle control applied to a foot, such as the position H-T describing the movement of stepping down on the heel then rising on the toe. Alternative footwork positions are H (heel), T (toe), T-H (toe-heel). The body turn may be specified as 1/2, 1/4, 3/4, 1/8, 3/8, 5/8 of a full revolution either to left or to right. The time beats used for a dance step may be 1, 2, 3, or 4 beats to reach the goal position of the step. More detailed description of these step positions are provided in Lake's thesis [Lake90].

Three dances are used in our examples, which are fox_trot, waltz, and test. Among them, the "test" dance is not one of the standard ballroom dances, but selected for the novice to practice the basic dance steps. The patterns in the "test" dance are a sequence of basic steps or a combination of several steps, such

as raising left and right arms, stepping forward, stepping backward, stepping forward and backward, and stepping to the side. For the "fox_trot" dance, six dance patterns are used in the examples, which are the backward_basic, box_step, closed_position_quarter_turns, forward_and_back_basic, promenade_basic, and twinkle patterns. Patterns selected from the "waltz" dance are box_step and left_box_turn. Note that the same box_step pattern is used in both "fox_trot" and "waltz" dances, but they have different time beats, footwork, and other modified control positions.

Consider one example of how the body segments are defined in the backward_basic pattern of the "fox_trot" dance. The foot positions in this eight-step pattern are defined, in order, as LF BWD (left foot backward), RF BWD, LF 2ND, RF 1ST, LF BWD, RF BWD, LF 2ND, and RF 1ST. These steps are defined by the standard positions and orientations of the body segments that can then be interpolated to produce a smooth motion. Motion from a set of positions and orientations to another set can be interpolated using several methods: linear interpolation, cubic interpolation, and interpolation using dynamic equations. Timing control of a movement between two keyframes can be handled while interpolating the movement. One example using the recursive dynamic equations of Armstrong and Green [Armstrong85] to interpolate two standard dance steps is given in Lake's thesis [Lake90]. His research also builds an interactive dance system which allows the user to interactively specify the standard dance steps and patterns.

The main focus of our research is to find a better control approach for motion in dynamic environments. The approach has intuitive, creative, and flexible control abilities, allowing motion to be easily generated and automatically adapted to a dynamic environment. As motion in a dynamic environment can be more variable, dependent, and individual than a single object's motion, its specification should directly address these difficulties. Motion can also be adapted to other behaviors and environments. Consider dance motion as one example. If two or more dancers are placed in a room environment, their dance motions are constrained by the room's boundaries, tables or chairs serving as static obstacles, other dancers serving as dynamic obstacles, and sound events in the room. How should the dance motion in this room environment be modeled independent of a sequential behavior and the environment? In other words, a behavior change

or an environment update only requires a minor revision of the modeled motion.

We use a set of relations to model the dance motion in this room environment. Standard dance steps used for the examples are generated from Lake's ballroom dance system. Motion sequences produced by our system consist of dance steps, other steps, body turns, pauses, and looking around. These output sequences can again be read into Lake's dance system for smooth motion interpolation. The main task of our system, as one example using the relation model, is to produce motion sequences (3D parametric key-frames) which naturally respond to a previously unknown dynamic environment. Examples using the system are incrementally developed from static environments to dynamic environments, from one dancer to several dancers, and from simple avoiding behavior to more complex environmental behaviors. Both relation descriptions and their hierarchical structures are introduced in each of these examples.

7.2. Dance Motion in Static Environments

A static environment is composed of one moving object and obstacles located in a restricted area. Here, two static environments are used: one with a group of objects and the one with additional individuals. These two examples cover the general issues of motion in static environments. In our examples, both the environments and motions in the environments are composed on-line in one interactive control session.

7.2.1. Static Environment with a Group of Obstacles

This example consists of a room environment with boundaries, one dancer, and a randomly placed group of blocks. One possible initial setting is to place the dancer in front of two blocks side by side and distribute the other blocks around the area. Each block can have a different size, color, and other properties. These, in turn, may trigger relations with various enabling conditions. In this room environment, we want to model the following behavior: the dancer selects a dance and pattern, and then dances the pattern in the room. Potential collisions with the walls or blocks are avoided in advance, by actions such as turning to a new direction. While avoiding potential collisions with one of the blocks, the dancer may show a dislike

feeling by a quick turning at the block (180 degrees), or a fear by using a few backward steps prior to turning to a new direction. These alternative behaviors of avoiding blocks can be used at different times, together with other behaviors in the environment to show the differences between individuals.

Five relations are used to model the behavior described above: "move_step", "avoid_walls", "avoid_blocks", "fear_a_block" and "dislike_a_block". "Move_step" advances the dancer to the next pattern step in the current dance, for the next keyframe instant. The pattern is repeated when the last step is reached. "Avoid_walls" produces an avoiding response whenever a wall is close to the dancer. This avoiding response closes the current pattern step and turns the dancer in a new direction for the next two steps. "Avoid_blocks" is similar to "avoid_walls", except the response is made to a block. "Fear_a_block" produces a fear response to a block that is larger than the others, or a particular block recognized by the dancer. This results in the dancer moving backward several steps when block is close to the dancer. "Dislike_a_block" results in a quick reverse turn when a block with a specific colour is too close to the dancer.

These relations as well as the modeling and rendering of the room, dancer, and blocks are described by the ORS language. A sample of these descriptions is included in Example One of Appendix A2. The source, responder, and enabling condition for each of these relations is summarized below:

relation	(source,responder):	enabling condition
move_step	(dancer,dancer):	motivated?
avoid_walls	(walls,dancer):	a_close_distance?
avoid_blocks	(blocks,dancer):	a_close_distance?
fear_a_block	(a_block,dancer):	distance_&_size?
dislike_a_block	(a_block,dancer):	distance_&_color?

The description of these relations and the corresponding objects is used to produce the system modules used as part of the interactive relation control system. The system supports interactive composition of the environment and the composition of relations to produce a desired behavior in the environment. For details of using the interactive system to compose an environment and motion, refer to Chapter 6. Here, we emphasize the structures necessary for a particular behavior and how these structures are built into an

environment.

Five relations are initially selected by the environment. The user can then interactively select two alternative relations, "fear_a_block" and "dislike_a_block", while modeling different behavior patterns. At the state control level, relations selected potential by both the environment and the user can be triggered to the active state when their conditions become true. When active, relations produce the following possible interactions: relation "move_step" is initially triggered active and advances the current dance step to the next one. At one point in the motion, "avoid_walls" is triggered active when its enabling condition, a close distance to one of the walls, becomes true. This relation issues a blocking state control to the "move_step" relation, which temporarily blocks the relation from producing a response during the avoiding period. A similar rule is applied when the relation "avoid_blocks" becomes active. If more than one block trigger this relation, the one with the shortest distance is selected for the current control step. This can be specified by assigning the distance values as the priority of the two relations.

The other two relations, "fear_a_block" and "dislike_a_block", are separately used in one of two patterns. Here we only look at the state control level. At this level, both relations are modeled with higher priorities than the relation "avoid_blocks", which enforces the fear behavior or dislike behavior prior to the avoiding behavior. For instance, at some point in the motion the relation "fear_a_block" is triggered active. In the active state, the relation issues a blocking state control to the avoiding relation, which is also triggered active by the block. This blocking structure allows the fear behavior be performed prior to the avoiding behavior. Similarly, the active relation "dislike_a_block" used in the other pattern issues a potentializing state control to the avoiding relation, which cancels the active avoiding behavior for the current control step. The canceled avoiding behavior may be triggered active at a later time, since the relation is only changed to the potential state.

Two behavior patterns called $p1$ and $p2$ are modeled in this example. The major difference between the two patterns is in the use of relations "fear_a_block" and "dislike_a_block", where the former is used in pattern p1 and the latter in pattern p2. The interactive state control among the relations for the two patterns

is outlined in Figure 7.2. These patterns are named and recorded in the system and later used as basic elements for modeling behavior sequences. These two patterns can be interactively selected, ordered, and scheduled, at the sequential control level to produce the desired sequential behavior.

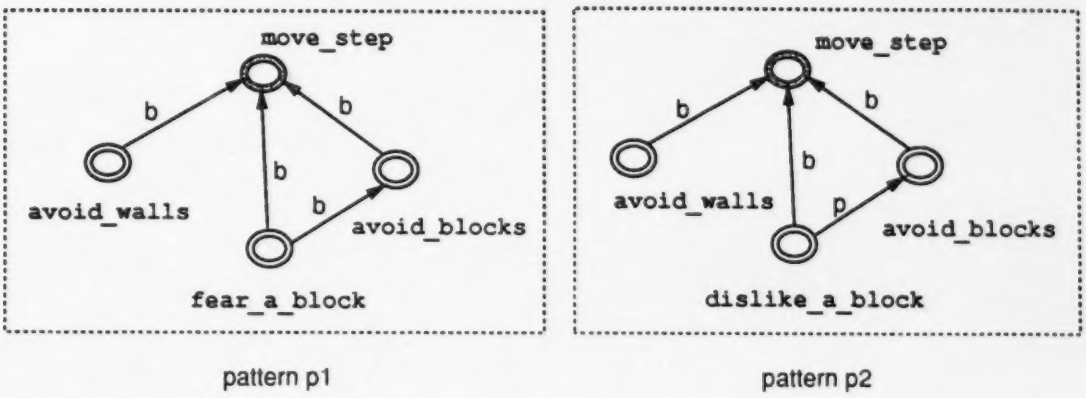


Figure 7.2 Interactive State Structures of Pattern P1 and P2

The following conventions are used in the figure. A relation is drawn by a double circle where the outside circle denotes the enabling condition and inside circle the response. An interactive state control is drawn by an arc pointed from the controlling relation to the controlled relation. Each state control type is indicated by a letter along the arc, where the letter "a" stands for an activating call, "p" for a potentializing call, "b" for a blocking call, and "t" for a terminating call. Notice the difference between the way that the "fear_a_block" and "dislike_a_block" relations interact with the "avoid_blocks" relation.

7.2.2. Static Environment with Other Individuals

Now, we extend the environment in the previous example by adding another object, a posting board. To model the motion in the extended environment, the object "posting board" and relations based on the board are first scripted using the ORS language, and then added to the interactive relation control system. When these descriptions are added to the previous example, the user can directly start from the previous example and extend the environment and motion behavior. To extend the environment, the user opens the scene composing component and interactively selects, locates, and includes the object "posting board" in the previously composed room environment.

Suppose the additional behavior based on the posting board is as follows: when the dancer sees the board for the first time, the dancer moves towards the board, reads the news on the board, and then dances away from the board. The next time the dancer moves close to the board, the dancer simply avoids the board and continues the dance. While avoiding the board, the dancer may decide to switch to a new dance pattern. Also, the dancer may switch to a new dance pattern after repeating the current pattern three times.

The additional behavior is modeled using the following additional relations: "to_board", "stop", "avoid_board", "change_newdance", and "change_newpattern". "To_board" moves the dancer towards the board, when the board appears in sight. "Stop" holds the dancer's motion for a short period. "Avoid_board" turns the dancer to a new heading away from the board for the next two pattern steps. "Change_newdance" changes the current dance. Finally, "change_newpattern" changes the current dance pattern to a new one, after it has been repeated three times.

"To_board" is mapped from the board to the dancer with the enabling condition that the board is in sight. "Stop" is mapped from the dancer to the dancer, with the default true condition, since the relation is called active by another relation. As the relation is not constrained by environmental conditions, it will begin responding when it is called active. "Avoid_board" is mapped from the board to the dancer, with the distance to the board as its condition. "Change_newdance" is mapped from and to the same object, the dancer, with the condition if it is called active. "Change_newpattern" is mapped also from and to the same

object, the dancer, with the condition that the current dance pattern has been repeated three times. The objects and enabling condition for each of these new relations are summarized in the following table. Detailed descriptions of these relations are found in Example Two of Appendix A2.

relation	(source,responder):	enabling condition
to_board	(board,dancer):	is_board_in_sight?
stop	(dancer,dancer):	if_called_active?
avoid_board	(board,dancer):	is_board_close?
change_newdance	(dancer,dancer):	is_called_active ?
change_newpattern	(dancer,dancer):	repeated_times ?

Assume all the new relations are initially selected by the environment. The second selection by the user divides these potential relations into two sets. One contains the relations "to_board", "stop", and "avoid_board", and the other has "avoid_board", "change_newdance", and "change_newpattern". This selection is based on the modeling of the two behavior patterns p1 and p2 in the previous example. At the state control level, new relations are used to produce the following interactions: "to_board" is triggered active if the board appears in the dancer's sight. While moving to the board, other relations such as "avoid_blocks" and "fear_a_block" may interact with the active relation "to_board", if they become active. One of these relations temporarily blocks the response towards the board until the response of avoiding the close block is complete. When the board is reached, the "to_board" relation sets the "stop" relation active, which produces a short pause at the board. The "stop" relation sets the "avoid_board" relation active at the end of its duration. This active relation produces an avoiding response at the board. This relation also issues a terminate call to the "to_board" relation to disable a second chance of moving towards the board, when the board again appears in dancer's sight. The next time the dancer moves close to the board, the "avoid_board" relation will be self triggered by its distance condition. The "move_step" relation is initially blocked by the "to_board" relation, but it is called active again at the end of "avoid_board" response. The moving relation advances the dancer to the next pattern step.

For the second relation set used for the behavior pattern p2, "move_step" is initially self triggered active to advance the dance pattern from one step to the next. Once the pattern has been repeated three

times, the "change_newpattern" relation is triggered active, which switches the current pattern to a new one. The dance motion could be blocked by the "avoid board" relation if the dancer moves close to the board. This active relation produces the necessary avoiding response at the board. At the end of the avoiding response, the "change_newdance" relation is called to change the current dance. The interactive controls amongst the new and old relations in the two behavior patterns, p1 and p2, are shown in Figure 7.3. A new notation in this example is the second letter enclosed in a pair of parentheses along the arc. This letter indicates the immediate state change of the interacting relation after issuing a state control. For instance, the letter (p) indicates the interacting relation's state is changed to potential right after issuing the call.

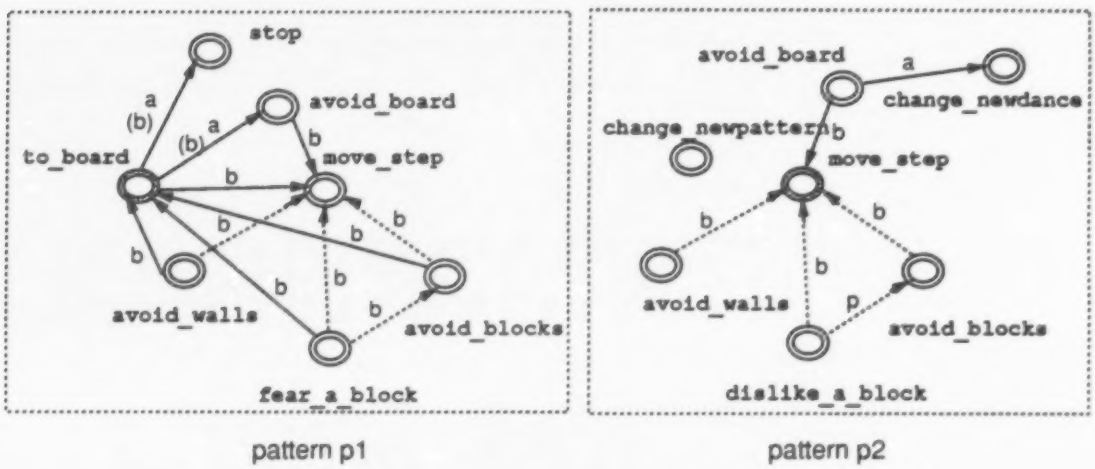


Figure 7.3 Revised State Structures by an Additional Individual

In the state control scheme shown above, the new control structures among the new and old relations are drawn in solid lines, while the previously used structures are drawn in dotted lines. One point to emphasize here is the call-active enabling condition. For the relation whose enabling condition is not defined, the relation will not be processed until it is called. This enabling condition is mainly used for common relations whose responses are usable in several similar situations, or the ones performing secondary

reactions, the less important reactions called by main reactions.

The two behavior patterns, p1 and p2, can be placed in different orders, starting from different times, and lasting for different lengths, while modeling a desired sequential behavior. One important feature of the relation model is the flexible motion revision based on simply changing the structures built on top of relations. For example, if the control of the relation "stop" is shifted from the "to_board" relation to the "avoid_board" relation in behavior pattern p1, a new behavior results, which shows a hesitant response after avoiding the board the first time. As the first experience towards the board shows a quick avoiding response to the board, the response gives a strong impression for the dancer's impatient character. The difference becomes more apparent if two dancers are placed in the environment, where one uses the same behavior pattern and the other uses the varied pattern. Similarly, a wide range of behaviors can be derived from the motion modeled by relations, such as shifting the control of the "stop" relation from behavior pattern p1 to behavior pattern p2.

7.3. Dance Motion in Dynamic Environments

Two types of dynamic objects can be involved in a dynamic environment: moving objects and scene events. Moving objects can be further divided into main characters and secondary (less important) characters. In our dance examples, the dancers are the main characters, and other objects such as a jumping triangle and a spinning stool are the secondary characters. Sound is the only scene event used in this example. These three dynamic objects are incrementally added to the environment, based on the block environment composed in the first example. The examples in the following subsections show the direct correspondence between the environmental and behavioral changes in the scene animation. The complexity of controlling the motion, as shown in the examples, only linearly increases with the number of relations and the structures imposed on them.

7.3.1. Dynamic Environment with Other Moving Objects

This example adds a jumping triangle to the block environment used in Example One. After this addition, the environment consists of walls, one dancer, a number of blocks, and one jumping triangle. The triangle's motion is controlled by an initial jumping direction, a constant speed, and a jumping distance. With these control properties, the triangle jumps step by step along the initial direction until it hits an object in the room. When this occurs, the triangle jumps in a new direction, bouncing back from the hit object. The dancer can also be hit by the jumping triangle. The only way to avoid the collision is to have the dancer sense the possible hit and take the necessary response to avoid the collision.

To model the additional behavior in the extended environment, five relations are used: "bounce_at_walls", "bounce_at_blocks", "bounce_at_dancer", "jump", and "avoid_triangle". The response of each new relation is described as follows. "Bounce_at_walls" assigns a new jumping direction to the triangle when it hits the wall. Similar responses are used for the relations "bounce_at_blocks" and "bounce_at_dancer", where the new jumping direction is assigned according to each hit instance. "Jump" moves the triangle to the next jumping step, along its assigned direction. "Avoid_triangle" advances the dancer by a few side steps to avoid a possible collision with the triangle.

The source and responder objects and enabling condition of each of the new relations are summarized as follows. For detailed description of the new object and the new relations in this example refer to Example Three of Appendix A2.

relation	(source,responder):	enabling condition
bounce_at_walls	(walls,triangle):	if_hit_by_wall?
bounce_at_blocks	(blocks,triangle):	if_hit_by_block?
bounce_at_dancer	(dancer,triangle):	if_hit_by_dancer?
avoid_triangle	(triangle,dancer):	close_triangle_insight?
jump	(triangle,triangle):	is_jumpable?

The environment now has two motions: the dancer's motion and triangle's motion. For the jumping motion, four relations are used: "bounce_at_walls", "bounce_at_blocks", "bounce_at_dancer", and "jump". Initially, these relations are selected by the environment. The state control of these relations, at the state

control level, is as follows: the "jump" relation advances the triangle to the next jumping step, if it is not disabled by other relations. If one of the relations, "bounce_at_walls", "bounce_at_blocks", or "bounce_at_dancer", is active, it disables the jumping motion, assigns a new bounce angle, and advances the first jumping step in this new direction. The jumping motion continues after the hitting response. The state control diagram among the relations for the jumping motion is shown in Figure 7.4.

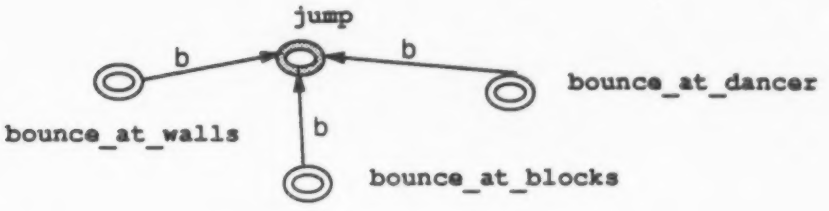


Figure 7.4 Interactive State Control of the Jumping Motion

For the dancer's motion, the only new relation is "avoid_triangle". This relation like the other avoiding relations may block the "move_step" relation when a potential collision with the jumping triangle is sensed. However, sensing a triangle collision may be ignored if the dancer is currently busy avoiding another obstacle, such as a block or a wall. This case shows a general conflict when two or more relations are active. The solution in such a case is to select the relation with the highest priority, assigned by either urgent or personal reasons. In this example, "avoid_triangle" is assigned a lower priority than the other two avoiding relations, "avoid_blocks" and "avoid_walls". Thus, a hit may not be avoidable if the dancer is avoiding a block or a wall. As a result, the triangle will hit the dancer and bounce back in another jumping direction. Other interesting behaviors can be produced if a different priority assignment is used. For instance, if "avoid_triangle" is assigned a higher priority than "avoid_blocks", an avoiding-hit response is used even if a close block is sensed. This unreasonable reaction may push the block or stop the motion at the block, which can be modeled using additional relations.

The interactive state control of the relations used in dance motion, in terms of the two behavior patterns p1 and p2, is shown in Figure 7.5. The same convention for drawing the graph, such as solid lines for new relations and dotted lines for old relations, is used here to show the additional control details.

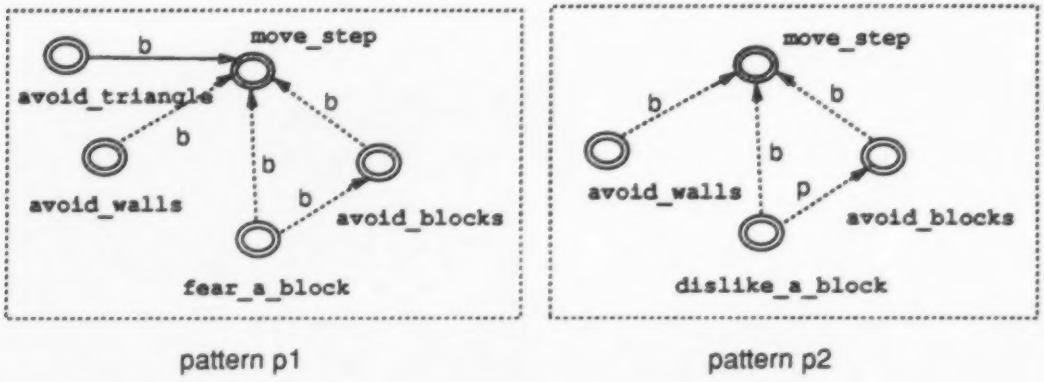


Figure 7.5 Revised State Controls with Other Moving Object

The two behavior patterns p1 and p2 are similarly structured and recorded as in the previous examples. One apparent difference between the patterns in this example is the use of the new relation "avoid_triangle". This new relation is only selected for pattern p1, whose interaction with other relations of the pattern is shown in the above diagram. Because the relation is used in pattern p1 and not in pattern p2, a different behavior, that concentrates on the dance and ignores the triangle, is produced by p2.

To stress the flexible editing of behaviors, consider the following behavior change. If relation "avoid_triangle" is replaced by relation "kick_triangle" (which produces a kicking response when the triangle is close to the dancer), a new behavior is produced by replacing this relation. This behavior suggests the dancer has an impatient character accordingly to his rude kicking attitude. Another example is to add an interactive control from the "bounce_at_dancer" relation to the "stop" relation, which generates a pausing response when the triangle hits the dancer. This additional behavior shows the sensitive character of the dancer, in comparison to the one without pausing. This additional change can be included in either pattern

p1 or pattern p2.

7.3.2. Dynamic Environment with Additional Dancers

This example adds two more dancers to the room environment. To produce this environment, two new dancers are interactively selected and assigned different height, width, and body segment ratios. Once selected, these dancers can be interactively placed in the room.

While dancing in the room environment, the three dancers can be identified based on their individual dance styles. For instance, dancer one independently chooses one of his favourite dances for practice; dancer two follows the dance pattern that dancer one is currently performing, being attracted by the graceful style of dancer one; dancer three, as a novice, repeats the basic steps of ballroom dancing. The three dancers, dancing individually, produce similar responses to other objects in the room environment, such as walls, blocks, and a jumping triangle. Between the dancers, the avoiding response is used first, which overrides other relations, such as attraction from dancer one to dancer two. A hit situation, however, may not be avoidable if two dancers are not facing each other.

Three new relations are used for the above behavior: "avoid_dancer", "hit_dancer", and "attract_dancer". The responses produced by these relations are as follows. "Avoid_dancer" closes the current dance step and then takes a few side steps to avoid a close dancer. If two dancers are responding to each other at the same time, both take the side steps in the opposite directions since they face each other. These responses quickly leave enough space between the dancers. Otherwise, if only one dancer is responding, he may need to take more side steps to successfully avoid the other dancer. "Hit_dancer" closes the current dance step and restarts the dance from the first pattern step. "Attract_dancer" rotates dancer two to keep dancer one in sight and switches his pattern to match the pattern that dancer one is using. In this case, dancer two normally changes his pattern whenever dancer one changes to a new pattern.

The objects (source and responder) and enabling condition for each of the new relations are outlined below. For details of the new relations refer to Example Four of Appendix A2.

relation	(source,responder):	enabling condition
avoid_dancer	(dancer_i,dancer_j):	if_too_close?
hit_dancer	(dancer_i,dancer_j):	if_hit?
attract_dancer	(dancer_1,dancer_2):	if_insight_&_pattern_match?

These new relations are initially selected by the environment. The user can again select the relations for each behavior pattern. Assume all the new relations are selected for modeling behavior pattern p1 and one relation, "hit_dancer", is selected for behavior pattern p2. The interactive state control using these relations in terms of dancer two's motion is as follows: "Move_step" initially advances dancer two to the next pattern step. "Attract_dancer" issues a blocking control to the moving relation when its enabling condition, dancer one is not in sight or dancer one has changed to a new pattern, becomes true. The active attraction relation produces a turn towards dancer one or a change of dance pattern. When relation "avoid_dancer" becomes active, it blocks either "attract_dancer" or "move_step", if either relation is active at that time. The same strategy is used for relation "hit_dancer" which, when triggered active, may block either the attraction relation or the moving relation. In the behavior pattern p2, relation "hit_dancer" adds the blocking structure to the move-step relation if a collision occurs. The interactive state control of relations in the two behavior patterns, in terms of dancer two's motion, is illustrated in Figure 7.6. This interactive control scheme can be similarly applied to the other two dancers' motions.

Interactive state control among the relations outlines the dynamic priorities assigned to the relations. In our example, the "attract_dancer" relation has a higher priority than the moving relation, and the "avoid_dancer" and "hit_dancer" relations have higher priority than both the attracting and moving relations. When more than one new relation is active at the same time, the one issuing the blocking control to the other relations is the one producing the active response. However, situations exist which can not be simply solved by using blocking control from one relation to others. One example is the avoiding relations. What happens when several avoiding relations are active at the same time? There is no unique solution for this question. The situation may become too complicated for the current relations to handle. One possible solution is to model a new relation which deals with this situation.

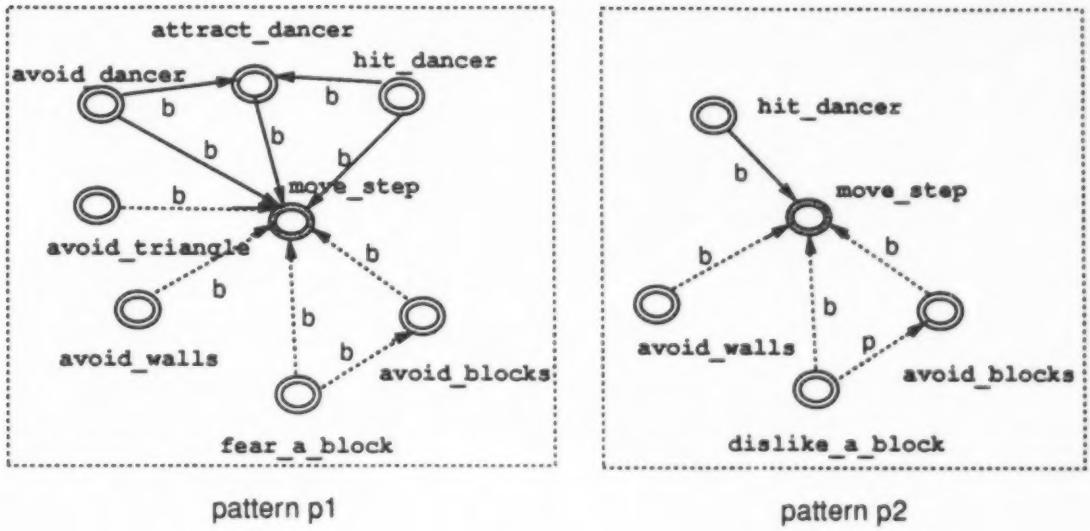


Figure 7.6 Revised State Controls by Two More Dancers

A simple solution to the multiple avoiding conflicts problem uses proper priority assignment. For example, if both relations "avoid_block" and "avoid_triangle" are active, relation "avoid_block" with a higher priority can issue a blocking control to the "avoid_triangle" relation, which may lead to a collision situation with the triangle. A similar case occurs between the two relations "avoid_block" and "avoid_dancer". If both are active, relation "avoid_block" with a higher priority can issue a blocking control to the other relation, which may lead to several hit-dancer responses during the avoiding-block response. If we reverse the priority assignment, by issuing a blocking control from relation "avoid_dancer" to the other relation, a hit-block situation might occur. The response in this situation can be modeled by a new relation which solves the crisis.

7.3.3. Dynamic Environment with Scene Events

The last example adds another dynamic source, a scene event, to the previous room environment. A scene event is treated as a special dynamic object source, which occurs in a scene at an unpredictable time and lasts for an indefinite period. Typical scene events are sounds, smells, lights, and birds passing over the scene. The event chosen for this example is a sound event. This is modeled using the control properties of time of occurrence, sound frequency, and sound period. Sound, as one type of dynamic object, can be selected and added to the room environment as any other object.

The additional behavior in response to the sound is as follows: The occurrence of a sound does not bother the jumping triangle as it may not be aware of the sound. Three of the dancers show individual attitudes towards the sound. When a sound occurs, dancer one stops normal dancing and looks around to search for the source. Dancer two keeps turning in the same place as a sign of surprise. Dancer three stops dancing and quietly waits for the sound to end. Once the sound event is over, all three dancers continue their dances.

This behavior can be modeled using three new relations: "look_at_sound", "turn_at_sound", and "stop_at_sound". "Look_at_sound" produces a looking around response and closes the current dance step while looking. "Turn_at_sound" closes the current dance step and rotates the dancer during the sound interval. "Stop_at_sound" does nothing but waits for the sound to end. These three relations have the same source object, the sound. A similar enabling condition is defined for the three relations: is the sound on? Based on the three relations, other alternative and interesting behaviors can be derived by simply exchanging the responder role of these relations. For instance, we can assign the looking relation to dancer three and the stopping relation to dancer one. We can change the "sound_on" condition to "sound_off" or "sound_on_2second", by a simple interaction. In the following, the objects and enabling condition of the new relations are outlined.

relation (source,responder): enabling condition

look_at_sound	(sound,dancer_1):	is_sound_on?
turn_at_sound	(sound,dancer_2):	is_sound_on?
stop_at_sound	(sound,dancer_3):	is_sound_on?

These new relations are initially selected by the environment in response to the new sound event. Each of the relations is added to one of the dancers' motions. Using dancer two's motion as an example, the new relation added is "turn_at_sound". Assume this new relation has higher priority than "move_step" and "attract_dancer", since the latter two are the most frequently used relations in dancer two's motion. This priority is explicitly structured in the state control diagram, where relation "turn_at_sound" blocks the active response of either relation "move_step" or relation "attract_dancer", while the sound event occurs. As the sound event can occur at any time, the response may need to block other relations currently active by using either a structure or a priority assignment. The extended state control diagram of behavior patterns p1 and p2 in terms of dancer two's motion is illustrated in Figure 7.7. These two patterns can be used in a sequential order p1→p2→ or p2→p1→.

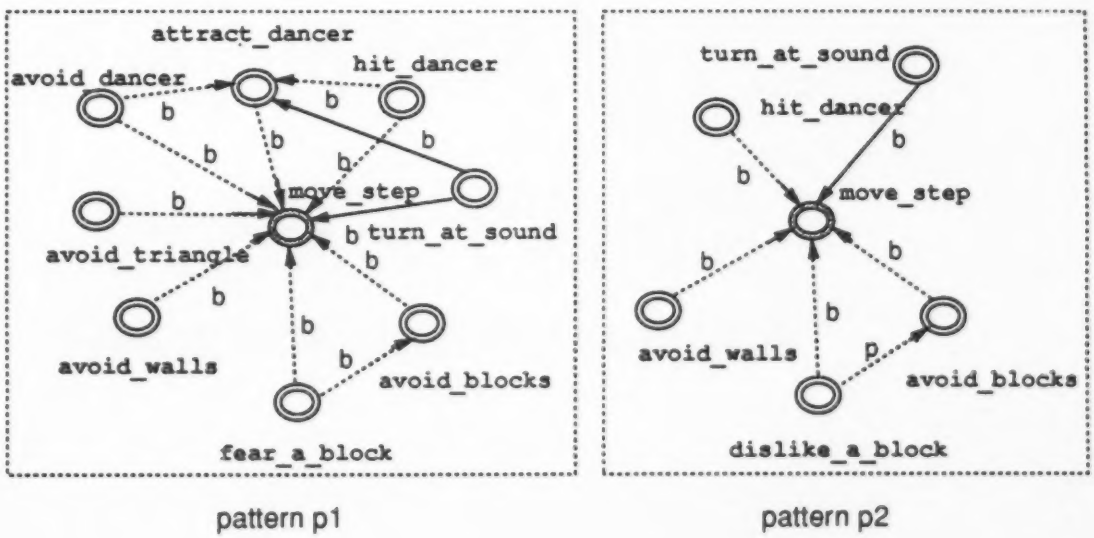


Figure 7.7 Revised State Controls with the Event Occurring

Another reason why the sound event response blocks one or all the other relations is the nature of the response to the sound. Suppose the response to the sound is modeled as a secondary reaction, such as nodding, clapping, or waving. This response can be overlapped with other relations when the sound event occurs. If the response to the sound is modeled as a main action, the active response should block other responses producing conflicting motions. In summary, the use of blocking produces a unique response, and the use of nonblocking produces a composite response. It can be true that the conditions sensed by another currently active relation may not allow the active response when an event occurs, such a condition could be a nearby block. In this situation, a new relation specially modeled to deal with it could be used.

Based on the modeling and structuring of relations, alternative behaviors can be derived by revising the relation control structure, adding a new relation, deleting an existing relation, and replacing an existing relation with a new one. Consider the three new relations used in this example. The deletion of "look_at_sound" shows the insensitive behavior of dancer one when the sound event occurs in the room. An extended use of "look_at_sound" for the three dancers produces an uniform behavior towards the sound. A role exchange between two relations, such as "look_at_sound" and "turn_at_sound", switches the behavior of the dancers. Other behaviors can be derived by varying each dancer's sensitivity according to the dancer's current like or dislike, or other environmental factors. These are a few of the alternative behavior examples using the three new relations. Together with the previously modeled relations, a wide range of environmental behaviors can be derived and explored in a conceptual and structural control mechanism.

Chapter 8

Conclusions

This chapter summarizes the main ideas proposed in the relation control model, along with its comparison with the other previous scene models. This chapter also outlines the major contributions of the research and its future work.

8.1. Research Summary

This thesis presents a new behavior control model, called the relation model, based on the special control issues of scene animation. This model produces a better control environment that encourages the intuitive, creative, and structured exploration of behaviors in dynamic environments. Working in this environment, the user is no longer restricted to predefined behavior sequences, but can freely express ideas about how a motion in an environment evolves.

Scene animation covers a wide range of environmental behaviors that have not been thoroughly studied in current animation research. As a result, the control concepts and mechanisms for the motion of a single object have been used for scene animation. It is possible to use techniques developed for a single object for a sequential behavior in an environment. But this can be very costly and time consuming because an animated sequence can not be generalized to another environment. Further, the motion in an environment can be more dynamic, dependent, and complex than the motion of a single object. These environmental factors increase the control complexity of the motion, which is not a linear addition of the behaviors of the objects in the environment, but some unknown exponential control function of the environment.

For a wide range of behavior applications, we would like to be able to easily change motion previously produced to handle alternative environmental behaviors. This minimum effort should be less than reprogramming the motion or redefining a set of keyframes. Using traditional approaches, a motion behavior is hardcoded into a program or a set of keyframes, which supports no structure for varying the behavior or the environment. This limited behavior modeling ability is one of the reasons why behavior

animation in a dynamic environment is unusually expensive.

Two important principles of our relation model are: the specification of atomic units of motion, and the ability to combine these atomic units to produce sophisticated behaviors. The environmental influences in a scene are decomposed into relations that describe how a single object responds to other objects and events in the environment. Each relation controls one environmental influence between two objects, the source object that triggers the motion and the responder that produces a response. The use of relations reduces the scene animation problem to smaller relation control problems.

Each relation can be modeled using a frame description. A relation, as a dynamic process, has its own sensor, response, duration, strength, and state. These control properties of a relation can be explicitly modeled in the relation frame and interactively adjusted while modeling the responsive behavior.

Relations once modeled are combined to form a relation composition hierarchy. This hierarchy consists of four control levels, which are called selective, state, pattern, and sequential. The selective level selects the relations used for a particular behavior application and environment. The state level specifies the interactive state controls among the relations. The pattern level combines relations that have the same behavior goal, and they can be used as identifiable behavior elements in the next control level. The sequential level orders the behavior patterns to form sequential behaviors. An ordered pattern list can call another list of patterns to form a branched control structure. Or, several ordered patterns or pattern lists can produce a looping control structure.

Using the relation composition hierarchy, the motion can easily be edited to produce other behaviors in the same or slightly different environments. For instance, the change can be as simple as changing the relations at the lowest level of the hierarchy, the selective control level. Other changes can be produced by revising the structures at higher levels, such as redirecting a state control, updating the relations in a grouping or scheduling, and switching the sequential order of two behavior patterns. These changes, based on the hierarchical control structure, are simple, explicit, and local in comparison with the changes in a program.

A prototype implementation of the relation control model has been produced. It contains two parts: the relation scripting language and an interactive behavior editor for composing the relations.

The language part of the system uses a textual, frame-like language to describe relations. A relation description is divided into three sections. The control header section of the frame is used for describing the local control properties of the relation. The action body section is used for describing the main control process of the relation, and the finalization section is used for describing the end control which needs to be separated from the main control body.

The behavior editor part of the system supplies an interactive control environment for the user to incrementally structure a sequential behavior, using the four hierarchical levels proposed in Chapter 5. The editor supports composing an environment, composing the behavior in the environment and revising the behavior to a wide range of alternatives.

A set of dance motions in a room environment have been produced using the prototype implementation of the relation control system. The dynamic dancing behavior in the examples are incrementally modeled in a series of environments ranging from simple static ones with a group of blocks and a postboard, to dynamic ones with jumping triangles, groups of dancers, and sound events. When the environment becomes more crowded with more dynamic sources, more relations are used to structure the natural behavior in the environment. Typical natural behaviors are to avoid collisions, express personal likes and dislikes, follow or assist other dance motions, and respond to other dynamic sources.

We feel that our initial research goal, to provide the user with an intuitive, creative, and structured control environment for scene animation, has been satisfied. First, the intuitive ability is supported by the decomposition of the motion into relations, which allows the animator to concentrate on one aspect of the motion at a time. The simplified control of relations matches the intuitive understanding of individual motion controls. Second, the creative ability is supported by the ability to model behaviors using a set of relations. Without dealing with the details, the user can concentrate on the behavior problem while using the relations as the basic elements. Third, the structured ability is supported by the hierarchical structures

explicitly modeled in the motion specification. The use of these structures facilitates changing the motion, a very difficult task if traditional programming is used.

One major difference between using the relation model and other previous models is the control environment provided for modeling and revising adaptive behavior in dynamic environments. The relation control environment limits the burden for specifying a complete motion to a set of relations, each of which describes one simple responsive behavior between two objects. In addition, the relation environment extends the interactive system's ability to the interface of hierarchical structuring mechanisms, based on the previously modeled relations. This extended ability allows the user to interactively define an environment as well as the behaviors in the environment in one interactive control session. Thus, such a system can be used as a general interactive control tool for exploring the potential behaviors and environments.

The use of a predefined-sequence model requires repeating the textual debugging cycle for any environmental or behavioral change. As no structure for supporting such a change is modeled in the code, the change can be made by repeating the cycle "program-compile-execute-view-reprogram", which can be very time consuming and costly. Using this model one pays the cost of one animated sequence for one unique behavior. For other slightly changed behaviors or environments, new sequences are modeled for each single case, each with detailed sequential coding.

The use of sensor-effector model relies on the understanding of how neural networks really work in humans and animals. Currently, the research in this area is still in its early stage. Very few results have been presented. It is still generally unknown how to model the motion of a realistic 3-D object, such as a human figure, in a dynamic scene environment.

The rule-based model outlines the concept of individual behaviors that should be modeled in a scene application. The use of this model presents a better view than the predefined sequence model, because the motion modeled by rules is independent of specific environmental locations, but a sequence isn't. However, the modeling and revising of behavior rules is still a low level activity. No higher control levels for rules is proposed at present for flexibly exploring similar environmental behaviors.

The predefined-environment model is mainly used for static environments. When this type of environment is extended to dynamic ones with multiple moving objects and unpredictable scene events, the basic assumption for using the model no longer exist. As for this reason, the techniques developed in the static scene domain may not be feasible for modeling the behavior in dynamic scene environments.

8.2. Major Contributions of the Research

The major contributions of this research are:

-- To our knowledge, this research is among the first to directly study the problem domain of scene animation. It is the first to formally discuss the problem, analyze its special control issues, and propose a solution, the relation control model, for the problem. The proposed relation model suggests an alternative (and superior) solution for the problem, in comparison with the predefined-sequence model, sensor-effector model, rule-based model, and predefined-environment model.

-- It addresses the difference between the motion of a single object and the motion of objects in an environment. Based on this difference, the theoretical foundation of the relation control model, including the relation concept, relation definition, relation class division, relation environmental mapping, and minimum and maximum number of relations required in an environment, is established.

-- The relation control model separates the control of scene animation into two steps: relation modeling and relation structuring. Both are generalization of particular control techniques and systems. The frame modeling of relations is independent of the use of particular control techniques. The relation composition hierarchy for structuring relations is independent of the use of interactive control devices available to a system.

-- The relation control model redraws the boundary between specification time and run time, as shown in Figure 8.1. Here, the specification time using the relation model is limited to relation modeling, and run time is extended to both behavior synthesis and editing. This means the interactive system interface is no longer restricted to issuing a motion command or adjusting a set of predefined

parameters. The use of the relation model opens the system's interface to a wider range of behavior applications.

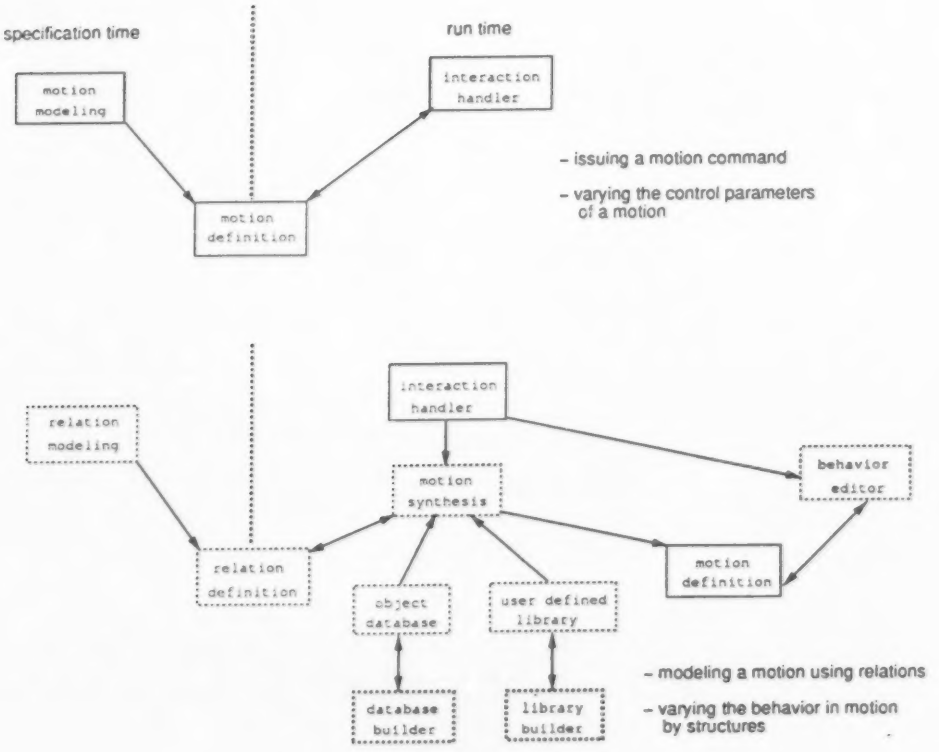


Figure 8.1 System Boundary Between Specification Time and Run Time

The relation control model has great potential for quickly and flexibly producing natural behavior in dynamic environments. This potential shows a significant step forward from the traditional and current scene animation approaches. The basic ideas proposed in the relation model can be applied to similar problems in education, robotics, medicine, and scientific simulation.

8.3. Future Work

As a first attempt, the relation control model is not necessarily complete and able to deal with every special scene case. The model, as proposed for the general scene problem, can be refined and expanded for other new demands. Potential improvement of the model will motivate continued research in similar directions. The following paragraphs contain suggestions for further research.

A better interface protocol between the levels of the relation composition hierarchy could be provided in a more natural way. A direct interface to specify the level control mechanisms among the relations could be developed based on a graphical display on the screen. With the relations displayed as circles (or other symbols), the user can select a relation by clicking the mouse on the relation's circle, link a state control between two relations by using a rubber band mouse interaction, and group relations with the similar behavior purpose by using mouse picking and dragging interactions. The strategies used for level structuring of relations can be included as part of the interface, and used as control options for the user's selection. A high-level or natural language interface at the top of the hierarchy could be implemented. If this is the case, the use of a hierarchy serves both bottom-up control of individual relations and top-down control of natural expressions, which certainly opens the relation control approach to a wider user community.

The idea of teaching by example commonly used in user interface research could be adopted to the relation control system. Users who are not familiar with the structuring mechanisms proposed by the relation model can learn from the examples demonstrated by the system. The system could offer both textual instructions and visual results of the motion in a clear way. The textual instructions could be organized in various levels of detail, to suit the users' understanding of the system. For the experienced users, however, this system feature could always be skipped.

One important reason for using alternative relations in a motion is to produce different motion experiences in the environment. Common experience is based on the number of occurrences of the same situation in the environment, such as avoiding obstacles. The current treatment of using alternative relations to model different experiences is left to the user. It is the user's role to arrange when and which relations should be

altered by the object's experience of moving in an environment. It would be better to let the system calculate the number of occurrences of similar situations and select alternative relations to vary the experience. In this way, the onus is not on the user to remember the experience and manually switch to alternative relations.

One research issue is to explicitly apply subjective control to an object's motion, such as an object's intention, mood, goal, and personal interests. The current system still models this type of control implicitly, using relation selection, state control, or grouping and scheduling structures. However, no explicit structure for modeling subjective control is outlined in the relation composition hierarchy. One suggestion is to provide an additional control level on top of the hierarchy to better organize the subjective control at the lower levels. This level should be able to supply the control necessary for a defined mood or correct the inconsistent errors in the mood at the lower levels.

Another suggestion for the future research is to adopt the decomposed relation control scheme to a parallel or a distributed computing environment. The use of such an environment could speed up the process of an animation using a large set of relations. One possibility is to use one machine as a master to interactively structure the relations, then distribute the relations along with the structures to other machines connected in a local network. When a structure is applied to a relation on another machine, a message simulating the structure could be constructed and delivered to the machine where the relation resides. A high-level network manager distributes the structured relations over the network and merges the results of the distributed controls.

References

[Armstrong79]

W. W. Armstrong, Recursive Solution to the Equations of Motion of an N-link Manipulator, *Proceedings Fifth World Congress on the Theory of Machines and Mechanisms*, 1979, 1343-1346.

[Armstrong85]

W. W. Armstrong and M. W. Green, The Dynamics of Articulated Rigid Bodies for Purposes of Animation, *Proc. of Graphics Interface 85*, May 1985, 407-415.

[Badler86] N. I. Badler, K. H. Manoochehri and D. Baraff, Multi-dimensional Input techniques and Articulated Figure Positioning By Multiple Constraints, *Proc. of Workshop on Interactive 3D Graphics*, 1986, 151-169.

[Badler89] N. I. Badler and B. L. Webber, Task-Driven Animation Using A Natural Language Interface, *Proc. of Workshop on Mechanics, Control and Animation of Articulated Figures*, April, 1989.

[Baecker69] R. M. Baecker, Picture-driven Animation, *Proc. Spring Joint Computer Conference 34*, (1969), 273-288, AFIPS Press.

[Baraff89] D. Baraff, Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies, *Computer Graphics 23*, 3 (1989), 223-232.

[Barr89] A. Barr, Teleological Modeling, *An Interdisciplinary Workshop on Mechanics, Control and Animation of Articulated Figures*, April, 1989.

[Barzel88] R. Barzel and A. H. Barr, A Modeling System Based on Dynamic Constraints, *Computer Graphics 22*, 4 (August 1988), 179-188.

[Braitenberg84]

V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, in *The MIT Press*, Cambridge, Massachusetts, 1984.

[Burtnyk76] N. Burtnyk and M. Wein, Interactive Skeleton Techniques for Enhancing Motion Dynamics in key Frame Animation , *Communication of the ACM 19*, 10 (1976), 564-569.

[Catmull79] E. Catmull, New Frontiers in Computer Animation, *American Cinema-tographer*, October, 1979.

[Chen88] M. Chen, S. J. Mountford and A. Sellen, A Study in Interactive 3-D Rotation Using 2-D Control Devices, *Computer Graphics 22*, 4 (August, 1988), 121-130.

[Coderre88] B. Coderre, Modeling Behavior in Petworld, in *Artificial Life*, Addison-Wesley Publishing Company, 1988.

[Crow88] F. C. Crow, Parallelism in Rendering Algorithms, *Proc. of Graphics Interface'88*, 1988.

[Csuri75] C. Csuri, Computer Animation, *Computer Graphics 9*, (Spring 1975), 92-101.

[DeFanti76] T. DeFanti, The Digital Component of the Circle Graphics Habitat, *Proc. of National Computer Conference'76*, 1976, 195-203.

[Esakov89] J. Esakov, N. Badler and M. Jung, An Investigation of Language Input and Performance Timing for Task Animation, *An Interdisciplinary Workshop on Mechanics, Control and Animation of Articulated Figures*, April, 1989.

[Featherstone83]

R. Featherstone, The Calculation of Robot Dynamics Using Articulated-body Inertias, *International Journal of Robotics Research 2*, 1 (1983), 13-30.

[Fisher86] S. Fisher, M. McGreevy, J. Humphries and W. Robinett, Vitual Environment Display System, *Proceedings of ACM Workshop on Interactive Graphics*, October, 1986, 77-87.

[Fishwick88]P. Fishwick, The Role of Process Abstraction in Simulation, *IEEE Transactions on Systems, Man, and Cybernetics 18*, 1 (Jan./Feb., 1988), .

[Forsey88] D. R. Forsey and J. Wilhelms, Techniques For Interactive Manipulation of Articulated Bodies Using Dynamic Analysis, *Proc. of Graphics Interface'88*, 1988.

[Fortin87] D. Fortin, J. F. Lamy and D. Thalmann, A Multiple Track Animation System for Motion Synchronization, *SIGGRAPH'87 Course Notes on Advanced Computer Animation*, 1987.

[Fournier82]A. Fournier, D. Fussell and L. Carpenter, Computer Rendering of Stochastic Models, *Communications of the ACM 25*, 6 (1982), 371-384.

[Girard85] M. Girard and A. A. Maciejewski, Computational Modeling for the Computer Animation of Legged Figures, *Computer Graphics 19*, 3 (1985), 263-270.

[Gomez86] J. E. Gomez, TWIXT: A 3-D Animation System, *Graphics Interface'86 Tutoral*, 1986.

[Grant86] E. Grant, P. Amburn and T. Whitted, Exploiting Classes in Modeling and Display Software, *IEEE Computer Graphics and Applications 6*, 11 (1986), 13-20.

[Green86a] M. Green and N. Bridgeman, *WINDLIB Programmer's Manual*, Univ. of Alberta, 1986.

[Green86b] M. Green, *FDB: A Frame Based Database System*, Univ. of Alberta, 1986.

[Green87] M. Green and J. Schaeffer, FrameWorks: A Distributed Computer Animation System, *Proc. CIPS'87*, 1987, 305-310.

[Green90] M. Green and C. Shaw, The Datapaper: Living in the Virtual World, *Proc. of Graphics Interface '90*, 1990, 123-130.

[Hahn88] J. Hahn, Realistic Animation of Rigid Bodies, *Computer Graphics 22*, 4 (1988), 299-308.

[Hanrahan87]

P. Hanrahan and D. Sturman, Interactive Animation of Parametric Models, *ACM SIGGRAPH'87 Tutoral on Computer Animation*, 1987.

[Honey71a] F. J. Honey, Computer Animated Episodes by Single Axis Rotations, *Proc. of 10th UAIDE Annual Meeting* , 1971, 3-120 to 3-226.

[Honey71b] F. J. Honey, Artist Oriented Computer Animation, *Journal of Society of Motion Picture and Television Engineers 80*, 3 (1971), .

[Horowitz83]R. Horowitz, Model Reference Adaptive Control of Mechanical Manipulators, *Ph.D. Thesis* , May 1983.

[Kajiya82] J. T. Kajiya, Ray Tracing Parametric Patches, *Computer Graphics 16*, 3 (1982), 245-254.

[Kajiya83] J. T. Kajiya, New Techniques for Ray Tracing Procedurally Defined Objects, *Computer Graphics 17*, 3 (1983), 91-99.

[Kalita89] J. Kalita and N. Badler, A Semantic Analysis of Action Verbs Based on Physical Primitives, *Proc. of Workshop on Mechanics, Control and Animation of Articulated Figures*, April, 1989.

[Karp90] P. Karp and S. Feiner, Issues in the Automated Generation of Animated Presentations, *Proc. of Graphics Interface'90*, May 1990.

[Korein83] J. Korein and N. Badler, Temporal Anti-aliasing in Computer Generated Animation, *Computer Graphics 17*, 3 (1983), 377-388.

[Lake90] R. M. Lake, Dynamic Motion Control of an Articulated Figure, *M.Sc. Thesis*, Spring, 1990.

[Lasseter87] J. Lasseter, Principles of Traditional Animation Applied to 3D Computer Animation, *Computer Graphics 21*, 4 (1987), 35-44.

[Lewis87] J. P. Lewis and F. I. Parke, Automated lip-synch and speech synthesis for Character Animation, *CHI+GI*, 1987, 143-147.

[Magnenat-Thalman83]

N. Magnenat-Thalman and D. Thalman, The Use of High-level 3-D Graphical Types in the Mira Animation System, *IEEE Computer Graphics and Applications 3*, 9 (December, 1983), 9-16.

[Magnenat-Thalmann85]

N. Magnenat-Thalmann and D. Thalmann, in *Computer Animation*, Springer-Verlag, 1985.

[Mandelbrot75]

B. B. Mandelbrot, Stochastic Models for The Earth's Relief, The Shape and Fractal Dimension of Coastlines, and The Number Area Rule for Islands, *Proc. National Academic Science USA 72*, 10 (1975), 2825-2828.

[Mandelbrot77]

B. B. Mandelbrot, *Fractals: Form, Chance and Dimension*, Freeman, San Francisco, 1977.

[Mandelbrot82]

B. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, San Francisco, 1982.

[McQueen87]

G. McQueen, Applying Classical Techniques to Computer Animation, *ACM SIGGRAPH'87 Tutorial on Computer Animation*, 1987.

[Moore88] M. Moore and J. Wilhelms, Collision Detection and Response for Computer Animation, *Computer Graphics 22*, 4 (1988), 289-298.

[Norton82] A. Norton, Generation and Display of Geometric Fractals in 3-D, *Computer Graphics 16*, 3 (1982), 61-67.

[Papert70] S. Papert, Teaching Children Thinking, *Proc. IFIP World Conference on Computer Education*, New York, 1970, 73-78.

[Parke75] F. I. Parke, A Model for Human Faces That Allows Speech Synchronized Animation, *Computers and Graphics 1*, 1 (1975), 1-4, Pergamon Press.

[Parke82] F. I. Parke, Parameterized Models for Facial Animation, *Computer Graphics and Applications 2*, 9 (1982), 61-68.

[Pars79] L. A. Pars, *A Treatise on Analytical Dynamics*, Ox Bow Press, Woodbridge, Connecticut, 1979.

[Platt88] J. Platt and A. H. Barr, Constraint Methods for Flexible Models, *Computer Graphics 22*, 4 (August 1988), 279-288.

[Potmesil82]M. Potmesil and I. Chakravarty, Synthetic Image Generation With a Lens and Aperture Camera Model, *ACM Transactions on Graphics 1*, 2 (1982), 85-108.

[Prusinkiewicz88]

P. Prusinkiewicz, A. Lindenmayer and J. Hanan, Developmental Models of Herbaceous Plants for Computer Imagery Purposes, *Computer Graphics 22*, 4 (August, 1988), 141-150.

[Reeves81] W. T. Reeves, Inbetweening for Computer Animation Utilizing Moving Point Constraints, *Computer Graphics 15*, 3 (1981), 263-269.

[Reeves83] W. T. Reeves, Partical Systems - A Technique for Modeling a Class of Fuzzy Objects, *Computer Graphics 17*, 3 (1983), 359-376.

[Reeves85] W. T. Reeves and R. Blau, Approximate and Probabilistic Algorithms for Shading and Rendering Structured Partical Systems, *Computer Graphics 19*, (1985), 313-322.

[Reynolds78]C. W. Reynolds, *Computer Animation In the World of Actors and Scripts*, SM thesis, Architecture Machine Group, MIT, 1978.

[Reynolds82]C. W. Reynolds, Computer Animation With Scripts and Actors, *Computer Graphics 16*, 3 (1982), 289-296.

[Reynolds87]C. Reynolds, Flocks, Herds and Schools: A Distributed Behavioral Model, *Computer Graphics 21*, 4 (July 1987), 25-34.

[Ridsdale88]G. Ridsdale, The Director's Apprentice: Animating Figures in a Constrained Enviornment, *Ph.D. Thesis*, 1988.

[Smith87] A. R. Smith, Formal Geometric Languages For Natural Phenomena, *ACM SIGGRAPH'87 Tutorial on Formal Geometric Languages*, 1987.

[Stern83] G. Stern, Bboop: A System for 3D Keyframe Figure Animation, *SIGGRAPH'83 tutorial*, 1983, 240-243.

[Takemura88]

H. Takemura, A. Tomono and Y. Kobayashi, An Evaluation of 3-D Object Pointing Using a Field Sequential Stereoscopic Display, *Proc. of Graphics Interface'88*, 1988.

[Thalmann84]

D. Thalmann and N. Magnenat-Thalmann, CINEMIRA: A 3-D Computer Animation Language Based on Actor and Camera Data Types, *Technical Report, University of Montreal*, 1984.

[Tomovic67]R. Tomovic, On Man-Machine Control, *Automatica 5*, 4 (1967), , Oxford.

[Travers88] M. Travers, Animal Construction Kits, in *Artificial Life*, Addison-Wesley Publishing Company, 1988.

[Ware88] C. Ware and D. R. Tessome, Using the Bat: A Six Dimensional Mouse For Object Placement, *Proc. of Computer Interface'88*, 1988.

[Wells69] D. A. Wells, Lagrangian Dynamics, *Shaum's Outline Series*, New York, 1969.

[Wilhelms5a]J. Wilhelms, *Graphical Simulation of the Motion of Articulated Bodies Such As Humans and Robots, With Particular Emphasis on the Use of Dynamic Analysis*, Doctoral Disertation, Computer Science Div., University of Califomial, July 1985a.

[Wilhelms87]

J. Wilhelms, Using Dynamic Analysis for Realistic Animation of Articulated Bodies, *SIGGRAPH'87 Course Notes on Computer Animation*, 1987.

[Wilhelms89]

J. Wilhelms and R. Skinner, An Interactive Approach to Behavioral Control, *Proc. of Graphics Interface'89*, 1989, 1-8.

[Witkin88] A. Witkin and M. Kass, Spacetime Constraints, *Computer Graphics* 22, 4 (1988), 159-168.

[Zeltzer81] D. Zeltzer and C. Csuri, Goal-Directed Movement Simulation, *CMCCS '81 / ACCHO '81*, 1981, 271-280.

[Zeltzer82] D. Zeltzer, Motor Control Techniques for Figure Animation, *IEEE Computer Graphics and Applications* 2, 9 (1982), 53-59.

[Zeltzer83] D. Zeltzer, Knowledge-Based Animation, *Motion: Representation and Perception*, April 1983, 187-192.

[Zeltzer85] D. Zeltzer, Towards An Integrated View of 3-D Computer Character Animation, *Proc. of Graphics Interface'85*, 1985, 105-115.

Appendix A1

Interactions in the System's Components

In the Scene Composition Component

-object selection

An object is selected from the object menu by two interactions: one is to activate the object and the other is to view the object in the test scene. An object is activated when the middle_button is clicked on the object item. A view of the object is produced when the right_button is clicked on the item. The right_button click action displays the object at the center of the test scene. Only the object previously activated (highlighted in the menu area), can be displayed in the scene. Remember the undo control effect of a middle_button click. A second middle_button click on an object item reverses the previous selection. In the case of a group object type, a right_button click on the object item adds one more object instance to the scene. As a group size is not fixed, the need for composing the group members depends only on the application. The most recently displayed object is called the active object.

-object resizing

An object in the environment can be interactively resized from the value menu. In the value menu, a set of graphical potentiometers are used to adjust the modeling parameters of an object. The range of a parameter value changes with the current value. The maximum value is always twice to the current value, which can be varied using mouse interactions. This extends the range of a parameter infinitely, rather than being restricted to a fixed range. For a group of objects, each active group member can be selected and has its own set of parameter values.

-object placement and orientation

The active object is initially displayed at the center of the test scene. This initial position can be changed by picking and dragging mouse interactions. A pick is initiated when the cursor is placed over an object and the left_button is pressed. While holding the button down, the picked object is dragged along with the mouse movement. This interaction sequence of picking and dragging can be

repeated several times until a satisfactory location is found. During this relocation process, the orientation of the active object can be changed by clicking on one of the rotate soft buttons. Each click on the button gives a quarter rotation of the object in either a clockwise or counterclockwise direction.

-object paste and cut

Once a final location for the active object is determined, the object can be added to the final environment by a mouse click on the "paste" soft button. To undo the paste action, a mouse click on the "cut" soft button is used. A cut is normally performed on the most recently pasted object. A cut can also be applied to a previously pasted object after a series of cuts have made it the most recently pasted object. A mouse click on the "showit" soft button displays the final environment. Each paste interaction adds an active object to the object linked list, while a cut interaction removes the last object on the list. A series of cuts may result in an empty list.

In the Pattern Control Component

The legal mouse interactions in the "group" mode are displayed in the on-line help area as follows: 1 groupat | 2 grouping | 3 cancel. This prompt says:

1 groupat

A left_button click selects the first relation to be grouped at a requested time step.

2 grouping

Consecutive middle_button clicks select other relations to be added to the group at the same time step.

3 cancel

A right_button cancels the current grouping structure.

When the left_button click occurs a dialogue box appears below the item name. To answer the question "group at time frame?" in the box, the user can type in a frame number. The box disappears as soon as an answer is received.

In the "schedule" mode, several relations that will execute at the same time are collected together. These relations will become active when another relation becomes active, during the active response of this relation, or when its active response terminates. The mouse interactions are prompted as "1 scheduleat | 2 scheduling | 3 cancel".

1 scheduleat

A left_button click selects the referenced relation for the group.

2 scheduling

Consecutive middle_button clicks select additional relations to be added to the group.

3 cancel

A right_button click removes the most recently scheduled structure.

The interaction for specifying one of the three time references is: as soon as the left_button click occurs on a relation item, a dialogue box appears with the message "select one time case?". For the cases of initial time and ending time, the box window closes after it receives the answer 1 or 3 from the user. If answer 2 is given, for an in-between time selection, a question "which in-between time slot?" is displayed in the box window. The question requests a specific time value relative to the initial time of the referenced relation. Once a time reference is selected, the middle_button clicks add more relations to the currently selected group.

The interactions between relations at each time step can be viewed in the display area. To facilitate viewing this structure, a set of soft buttons are used in the area. Three appear at the upper-right corner of the area: "addone", "backone", and "backall".

addone

A mouse click on the "addone" soft button inserts a marker in the current motion frame.

backone

A click on the "backone" button reverses the recorded motion to the last inserted marker, then a new motion can be constructed.

backall

A click on the "backall" button resets the motion to the beginning, with all the markers removed.

The use of these buttons gives a flexible environment for modeling a pattern structure. This is useful in determining things such as the best time for a grouping or scheduling and which is the most suitable relation to use as a reference.

The use of the mouse buttons in the display area is given by: "mouse: 1_initial 2_recording 3_playback", in the lower-right corner of the area. These three functions define the way a motion can be interactively recorded and played back.

1_initial

Each left_button click in the area, except the area covered by the soft buttons, starts a new motion. The new motion deletes any previous motions and starts the first run of recording (12 frames).

2_recording

Consecutive middle_button clicks continue the recording for 12 more frames.

3_playback

A right_button click reviews the whole recorded motion.

Using these mouse functions, the user can record a motion and see how the motion progresses in the environment for some period of time, with an adjustable playback rate (30 frames/second is initially assumed).

While a recorded motion is displayed frame by frame, the two pattern control structures, grouping and scheduling, are also dynamically drawn in the relation diagram together with other state control structures. A grouping structure is drawn with a dotted line pointing from a grouped relation to the center of the relations circle, with the grouped frame number by the end of the line. A dotted line is also drawn for a scheduling structure, from a scheduled relation to the referenced relation. The three types of scheduling are indicated with the numbers (1,2,3) along each line. One of the lines, either a grouping or a scheduling, is drawn when the structure it represents becomes active. Otherwise, the line will not be drawn in the current time step. The dynamic display of control structures during a motion's progress ties the motion to the relevant modeled structures.

In the Sequential Control Component

The two control modes for extending the basic environment to a general environment are:

picking

The picking mode is entered after a mouse click on the "pick" button. In this mode the user selects an object by pointing at it and pressing a mouse button. A picking interaction duplicates the picked object by creating a new instance. As soon as an object is picked, the picking mode is automatically converted to the moving mode.

moving

In the moving mode, the new object instance follows the mouse movement while the mouse button is pressed.

Switching modes is indicated by a highlighted button.

The dragging of a picked object can be repeated more than once, as long as the interface is in the moving mode. There are three ways of leaving the moving mode. One is a mouse click on the button "move", which toggles the mode. The second is a mouse click on the "pick" button, which selects the picking mode. The third is a mouse click on the "cancel" button, which deletes the picked object from the general environment and at the same time cancels the active moving mode.

Once the general environment is composed, a sequential behavior is produced by interactions in the menu area. Mouse functions used in the "pattern" menu are "1 timemenu | 2 activeitem | 3 quicktime". The "pattern" menu shows the pattern names previously created.

1 timemenu

A left_button click on a pattern item traverses the menu hierarchy to the "time" menu.

2 activeitem

A middle_button click on a pattern item makes the pattern active. This selection is canceled by another middle_button click. Selection is one criterion for a pattern's participation in a motion

sequence. Another criterion is the time scaling of the pattern.

3 quicktime

A right_button click on a pattern item expands the time duration of the pattern by 10 frames.

The "time" menu contains two time variables f1 and f2 for its parent pattern. These two variables mark the time boundary of the pattern in the motion sequence. The pattern is active when it is selected and its starting time, f1, matches the motion recording frame. An active pattern continues until its ending time given by f2 is reached. The use of a pattern causes a structure copy and remove process. When a pattern is active, the control structures among the participating relations, are copied to the system control environment. These structures are removed from the system when the pattern ends.

The "time" menu is used to specify the value for either f1 or f2. Three button interactions are used to set these values. A left_button click in either the f1 or f2 slider sets a new value. A middle_button click in the slider decreases the value by 5, and a right_button click increases the value by 5.

Sequential structures among patterns can be viewed from the system message area. Each time a new ordering or a new time slot is specified, a new pattern list is shown in the message area. From there, the user can easily see how patterns are ordered sequentially, possibly in parallel as well, and the time connection between patterns. Interactions used for viewing a structured sequence are mainly produced in the display area. These interactions are based on the following mouse actions "1_initial | 2_recording | 3_playback", in the lower right corner of this area. The length of the recording cycle, started by clicking either the left_button or middle_button, is determined by the minimum value f1 and the maximum value f2 for all the active patterns.

1_initial

A left_button click signals a new recording cycle and removes the previous recording (if any).

2_recording

A middle_button click continues the recording of the next cycle. A continuous recording is possible

if a new active pattern is added.

3_playback

A right_button click plays back the recorded motion sequence.

During the experiments, any blank time steps left between two adjacent active patterns are ignored by the system. In this case, the last generated frame is copied for each time step, which shows static images when the motion is displayed. The appearance of static images shows the missing time connection between patterns.

In the Scene Rendering Component

The steps for mapping a selected sample color to a color panel are:

selecting a panel

The first step is done by using the *forward, backward, newpage, and backpage* commands, which moves the index pointer to an available color panel. If a new page of color panels or a previous page is selected, the index pointer is always placed at the first color panel of the page.

copying color

The second step is to copy the sample color to another panel using the *copy* command.

Colors mapped to the panels are the ones used for the color mapping of the system. The use of these colors are indexed by the panel number, which may be part of an object's modeling properties.

A previously recorded motion sequence can be interactively reviewed in a selected camera mode, in terms of the three mouse button functions. These functions are: "1_initial 2_stopping 3_stepping", as prompted in the lower-right corner of the display area. This message says:

1_initial

A left_button click in this area starts a continuous display of the motion, at a default display rate.

2_stopping

A middle_button click stops the display.

3_stepping

A right_button click advances the motion one frame at a time. The stepping display automatically circles from the last frame to the first frame when the end of motion is reached.

Between the stepping intervals, different camera modes can be selected.

Appendix A2

Modeling of Relations in Various Environments

Example One: In a Room with Obstacles

This is an example for modeling the objects and relations used for the dance motion, in a room environment with a group of obstacles such as blocks.

```
Scene_modeling
  Poly_class

    object name: walls
    type: POLYGON
    color: 32
    vertices: -5.0 0.0 -5.0 ,  5.0 0.0 -5.0 ,
              5.0 0.0  5.0 , -5.0 0.0 5.0 ;

    object name: floor
    primitive: {
            struct stripes {
              double swidth;
              int scolor; };
            struct stripes floor;
           }
    para_def: double sw 0.5 , int sc 33 ;
    initial: {
            floor.swidth = sw;
            floor.scolor = sc;
            }

    object name: block
    primitive: {
           struct blockobj {
           . . . /*  block model definition };
           struct blockobj block[];
            }
    para_def: double sx 0.1 , sz 0.1 ,
              . . . /* initial control values
    initial: {
              . . . /* assignment of initial values
             }

    object name: dancer
    primitive: {
            struct dancers {
              . . . /* dancer model definition };
             struct dancers dancer[];
             }
```

```
    para_def:  int sp 1 , sr 2 ,
               ...
    initial: {
               ...
             }

Scene_motion
  relations

    /* relation "avoid_walls"
    object name: dancer
    with: walls
    visual sensor: atonewall(dancer.x+dx,dancer.z+dz)
      ... /* response parameters
    initial: Active
    avoid_walls {
        ... /* local control parameters

        teststep(ndancer,&dx,&dz);
        visual sensor: {
          setactive();
          if( first_time ) {
            Callactivefrom();
            CallDeactive(move_step);
            ... /* initial control } }
        if( sensing_is_true ) {
          Callactivein();
          if( !savetogo(dancer) )
            turndancer(ndancer); }
        else {
          CallActive(move_step);
          Callactiveto();
          setdeactive();
          ... /* other ending control }
        }

    /* relation "dislike_a_block"
    visual sensor: atoneblock(...) && Nblock==dblock
    para_turn: int dblock 3 ;
               ...
    initial: Active
    dislike_a_block {
        ... /* local control parameters

        teststep(ndancer,&dx,&dz);
        visual sensor: {
          ... /* initial control }
        if( sensing_is_true ) {
          ...
          /* a dislike response }
```

```
        ]
      => {
         if( avoided_block ) {
            CallActive(move_step);
            ... }
         }

/*  relation "avoid_blocks"
with: block
visual sensor: atoneblock(...)
para_turndir: int signs -1 <-> 1 ;
    ...
initial: Active
avoid_blocks {
     ... /* local control parameters

     teststep(ndancer,&dx,&dz);
     visual sensor: {
        ... /*  initial control }
      if ( sensing_is_true ) {
           ...
         /* an avoiding response }
        }
   =>  {
        if( avoided_block ) {
           CallActive(move_step);
           ...     }
         }

/*  relation  "fear_a_block"
visual sensor:  atoneblock(...) && Nblock==fblock
para_range:  double atrange 0.3 , int flock 2 ;
        ...
initial: Active
fear_a_block {
    ... /* local control parameters

    teststep(ndancer,&dx,&dz);
    visual sensor: {
       CallDeactive(avoid_block);
       ... /* initial control  }
     if( sensing_is_true ) {
       ...
       /*  a fear response  }
      ]
   => {
      if( avoided_block ) {
         CallActive(avoid_block);
         ... }
       }
```

```
        /*  relation "move_step"
        visual sensor: ismovable(ndancer)
        initial: Active
        move_step {

                visual sensor: {
                  setactive();
                  advanceonestep(ndancer); }
                else setdeactive();
                }

Scene_render

      floor {
            drawfloor(Scene);
            }

      block {
            drawblock(nblock,Scene);
            }

      dancer {
            drawdancer(ndancer,Scene);
            }
```

Example Two: With Additional Static Individuals

This is an example for modeling an additional board object and relations triggered by the board as the sources. The example only shows the modeling code extended from the previous example.

```
object name: board
primitive: {
   struct boardobj {
     . . . /* board model definition };
   struct boardobj board;
    }
para_def: double sx 0.1 ,
     . . . /* initial control values
initial: {

     board.x = sx;
      . . . /* assignment of initial values
    }

/*  relation "change_newpattern
initial: Active
visual sensor:  ifnewpattern(...)
change_newpattern {

     visual sensor: {
       setactive();
       nextpattern(ndancer); }
     else
       setdeactive();
  }

/*  relation "stop"
para_time: int vst 3 <-> 5 ;
stop {

     if( clock == 0) {
        setactive();
        clock = vst;
        CallDeactive(move_step); }
     clock--;
     if( clock == 1) {
        CallActive(move_step);
        setdeactive();
        clock = 0;    }
      }

 /*  relation "change_newdance"
 change_newdance {
```

```
    if( first_time ) {
      restartpattern(ndancer);
      . . . /* switch to a new dance } }
    else setdeactive();
  }

/*  relation "to_board"
with: board
initial: Active
visual sensor: isboardinsight(ndancer)
to_board {
    ... /*  local control parameters

    visual sensor: {
      if( first_time) {
        . . .
        CallDeactive(move_step);
        CallDeactive(avoid_board);
        . . .     } }
     if( board_is_insight) {
        . . .
        if( not_face_to_board)
          adjusttoboard(ndancer);
        else if( !too_close || !close_step)
          forwardstep(ndancer,..);
        else if( close_step )
          closestep(ndancer);
        if( too_close )
          CallActiveBlock(stop);
        . . .
        if( end_response) {
          . . .
          CallActive(move_step);
          CallActive(avoid_board);
          . . .    } }
       }

/*  relation "avoid_board"
with: board
visual sensor: atboard(dancer.x+dx,dancer.z+dz)
initial: Active
avoid_board {
    . . . /*  local control parameters

    teststep(ndancer,&dx,&dz);
    visual sensor: {
      if( first_time) {
        setactive();
        CallDeactive(move_step);
        . . . /*  initial control } }
```

```
    if( sensing_is_true ) {
        . . .
      if( intersectboard(..) )
        rotatedancer(ndancer);
       . . .          }
    else {
       setdeactive();
       CallActive(move_step);
        . . .     }
     }

/* rendering
board {
       drawboard(Scene);
       }
```

Example Three: With Other Moving Objects

This is an example for modeling an additional jumping triangle object, as another moving object, in the room and relations used in the jumping motion as well as the dance motion. The relations extended for the dance motion are due to the presence of the triangle in the environment.

```
object name: triangle
primitive: {
     struct sideobj {
       . . . /* triangle model definition };
     struct sideobj triangle;
   }
para_def: double sx 0.1 ,
    . . . /* initial control values
initial: {

    triangle.x = sx; . . .  /*  assignment of initial values }

/*  relation "jump"
object name: triangle
initial: Active
para_size: double vh 0.4 ;
jump {
    . . . /* local control parameters

    setactive();
    jumpstep(&dx,&dz);
    . . .
   }

/*  relation "bounce_at_block
with: block
initial: Active
visual sensor: atoneblock(triangle,Nblock)
bounce_at_block {

    visual sensor: {
     if( first_time) {
        setactive();
        changeheading(triangle); }
     else setdeactive(); }
   }

/*  relation "bounce_at_dancer
with: dancer
initial: Active
visual sensor: atonedancer(triangle,dancer)
bounce_at_dancer {
```

```
    visual sensor: {
     ...
     /*  a bouncing response }
  }

/*  relation "bounce_at_wall
with: walls
initial: Active
visual sensor: atonewall(triangle)
bounce_at_wall {

    visual sensor: {
     ...
     /*  a bouncing response  }
  }

/* relation "avoid_triangle
with: triangle
visual sensor: isinsight(...) && hittriangle(...)
initial: Active
avoid_triangle {
   ... /*  local control parameters

   teststep(ndancer,&dx,&dz);
   visual sensor: {
     if( first_time ) {
        setactive();
        CallDeactive(move_step);
        restartpattern(ndancer);
          ...    } }
   if( sensing_is_true ) {
      ...
      if( hittriangle(triangle,ndancer) )
        onesidestep(ndancer)
      else if( close_step)
        closestep(ndancer)
      ...     }
   else {
        setdeactive();
        CallActive(move_step);
          ...    }
    }

   /*  rendering
   triangle {
         drawtriangle(Scene);
         }
```

Example Four: With Additional Dancers

This is an example for modeling relations extended from the room environment with several dancers. Because the dancer object is declared as a group, no additional code for modeling and rendering the additional dancers is necessary.

```
/*  relation "avoid_dancer
object name: dancer
with: dancer
visual sensor: atonedancer(ndancer,Ndancer)
initial: Active
avoid_dancer {
    ...  /*  local control parameters

    if( ndancer == Ndancer)
      continue;
    teststep(ndancer,&dx,&dz);
    visual sensor: {
      if( first_time) {
        setactive();
        restartpattern(ndancer);
        lockdancer(ndancer);
         ... /* initial control  } }
     if( ndancer_has_true_sensing) {
        ...
       if( atonedancer(ndancer,Ndancer))
         onesidestep(ndancer);
       else if( close_step)
         closestep(ndancer);
       ...      }
      }
   ==> {
      if( ndancer_ends_response) {
        unlockdancer(ndancer);
        setdeactive();
         ...  }
        }

   /*  relation "hit_dancer
   with: dancer
   visual sensor: hitonedancer(ndancer,Ndancer)
   initial: Active
   hit_dancer {
      ... /* local control parameters

      if( ndancer == Ndancer)
        continue;
      visual sensor: {
```

```
    if( first_time) {
       setactive();
       clockdancer(ndancer);
       restartpattern(ndancer);
         ...  } }
  if( ndancer_is_hit ) {
     setdeactive();
     unlockdancer(ndancer);
     ...     }
   }

/* relation "attract_dancer
with: dancer
visual sensor: !insignt(...) ||
          ifvarypattern(ndancer,Ndancer)
initial: Active
para_parts: int sd 1 , rd 2 ;
attract_dancer {
   ... /* local control parameters

  if( Ndancer!=sd || ndancer!=rd)
     continue;
  visual sensor: {
   if( first_time) {
      setactive();
      lockdancer(ndancer);
      copypattern(ndancer,Ndancer);
      ...       } }
   if( ndancer_has_true_sensing) {
      ...
      getturnto(ndancer,Ndancer); }
      ...
   else {
        setdeactive();
        unlockdancer(ndancer);
        ...     }
     }
```

Example Five: With Additional Sound Events

This is an example for modeling an additional sound event and relations in response to the sound, based on the previous room environment. Here, the sound event is procedurally modeled as a dynamic object and its appearance is identified by a period of sound in the environment.

```
object name: sound
primitive: {
   struct soundobj {
     int timeon;
     int period;  };
   struct soundobj sound;
  }

/*  relation "look_at_sound
object name: dancer
with: sound
sound sensor: issoundon() && ndancer==pickd
initial: Active
para_dindex: int pickd 2 ;
look_at_sound {
   . . . /*  local control parameters

   sound sensor: {
     if( first_time ) {
        setactive();
        lockdancer(ndancer);
        CallDeactive(avoid_dancer);
          . . .     }
      else turnhead(ndancer);   }
    else {
         setdeactive();
         unlockdancer(ndancer);
         CallActive(avoid_dancer);
         . . .       }
    }

  /*  relation "turn_at_sound
  with: sound
  sound sensor: issoundon() && ndancer==pickd
  initial: Active
  para_dindex: int pickd 3 ;
  turn_at_sound {
     . . .  /*   local control parameters

     sound event: {
       if( first_time) {
          setactive();
```

```
        lockdancer(ndancer);
        ...
        clock = 6; } }
    if( --clock > 0)
      turndancer(ndancer);
    else {
        setdeactive();
        unlockdancer(ndancer);
         ...  }
    }

/* relation "stop_at_sound
with: sound
sound event: issoundon() && ismovable(ndancer)
initial: Active
stop_at_sound {
    ... /* local control parameters

    sound event: {
       lockdancer(ndancer);
       if( first_dancer) {
          setactive();
          CallDeactive(avoid_dancer);
          ...
          CallActiveBlock(stop);
          ...  } }
       if( ndancer_locked) {
          unlockdancer(ndancer);
          if( first_dancer) {
             setdeactive();
             CallActive(avoid_dancer);
               ...      } }
         }
```